WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

EFMD
EQUIS
ACCREDITED

Master Thesis

Business Applications of Quantum Machine Learning

Harald Zeindlinger

Date of Birth: 25.07.1998
Student ID: 11814612

Subject Area: Digital Economy

Studienkennzahl: UJ 066 960

Supervisor: Univ. Prof. Dr. Axel Polleres

Date of Submission: September 1, 2024

Department of Information Systems & Operations Management, Vienna University of Economics and Business, Welthandelsplatz 1, 1020 Vienna, Austria

DEPARTMENT FÜR WIRTSCHAFTSINFORMATIK UND OPERATIONS MANAGEMENT DEPARTMENT OF INFORMATION SYSTEMS AND OPERATIONS MANAGEMENT

Contents

List of Figures

List of Tables

Abstract

Quantum computing offers a fundamentally different approach to information processing than classical computing. With data encoded as quantum states, quantum algorithms can achieve significant speedups in cryptography, search, and linear algebra. Although current hardware limitations prevent realization of these capabilities, recent experiments suggest potential utility of gate-based quantum devices even without full error correction. One candidate for such utility is Quantum Machine Learning (QML), which is directed at building hybrid quantum-classical workflows for learning tasks. This thesis explores the applicability of such techniques in typical business contexts from common Data Science scenarios. To this end, it first establishes a mapping of use cases and classical ML methods by reviewing previous Data Science Lab projects at WU Vienna. After introducing basic principles and current state of prominent QML methods, a review of existing works on their business applications is conducted. Finally, the thesis presents two proxy cases implemented with publicly available datasets on churn prediction and predictive maintenance.

1 Introduction

In the age of the Digital Economy, Data Science and Machine Learning (ML) methods have become indispensable tools for businesses across domains to drive value creation. While Data Science encompasses the overall workflow from turning data into problems to finally deriving action and building products, ML algorithms are at the center of this process: From gaining insights about potential and current customers, over optimizing processes, to designing, deploying, and evaluating new products and services, all steps in the business life-cycle can potentially benefit from their application [72]. More recently, the emergence of Deep Learning (DL) has led to even more steeply growing corporate adoption of such techniques, empowered by the combination of massive data availability, increased computational power via Graphical Processing Units (GPUs), and novel algorithmic approaches [26]. Furthermore, due to the not yet fully understood phenomenon of scaling laws, the performance of DL architectures like transformers or convolutional neural networks (CNNs) can be improved by increasing the number of model parameters and the size of the training data, as opposed to architectural changes [9].

However, this current state of ML is certainly not the end of history in the evolution of data-driven business, in the sense that its powerful algorithms can or should be leveraged as default solutions to any conceivable challenge in the enterprise. This is due to three main reasons, two of which form the starting points for this thesis. Firstly, the availability and quality of data naturally limits the insights achievable for Data Science teams and therefore constitutes a central precondition for their endeavors. Secondly, the way of asking a question about the data determines the nature and quality of the results: ML algorithms operate on numerical representations of (real-world) entities, but there seldom exists only one way of defining these representations. This task of problem formulation is located in the early stages of Data Science process models - in the CRISP-DM standard, for example, it emerges from the interplay of business and data understanding [109]. Reconsidering problem formulation in every iteration of the process is crucial for aligning statistical analyses with the actual business needs, including the imposed legal and ethical restrictions. Clearly, this calls for explicit, careful decisions on desired outcomes and selection of algorithms suiting the given problem, instead of blind reliance on conveniently available tools [77]. Thus, proven strategies and best practices are desirable to support corporate Data Science teams in taking accountable decisions, an aspect which will be reflected in our **first research question**.

The third major limitation on utilization of current ML and DL methods is posed by the immense computational effort required for model training. This technological issue comprises of many aspects, such as long training duration, high energy consumption, the high amount of data needed to learn hidden patterns, as well as the question whether the employed resources ultimately lead to good generalization of the model on unseen data. All of these points, which also constitute considerable cost factors, indicate possible definitions of where to search for "better" ML models, compared to the current state of the field [91]. Given recent technological advancements and increased availability of quantum computing resources, Quantum Machine Learning (QML) is now emerging as a future paradigm for potentially achieving these desired improvements. In general, the hypothesized benefit of ideal quantum computers rests on three pillars: They can solve classically intractable problems such as prime factorization, they offer speedups for certain classically solvable problems, and they cannot be fully simulated with a classical computer [82]. While current quantum hardware still belongs to the NISQ (Noisy Intermediate Scale) era, characterised by errors arising from the physical implementation, experimental results have demonstrated that quantum computers are capable of outperforming state-of-the-art classical supercomputers: The Google Sycamore quantum device completed an experiment with theoretical classical runtime of 10.000 years within 200 seconds [7]. Furthermore, the most recent IBM superconducting processors were able to deliver accurate expectation values for tasks where the best known classical simulations break down, therefore yielding evidence for practical utility of quantum computing even before achieving full resilience against errors [49]. These results exemplify the motivation for leveraging the characteristics of quantum information and the capabilities of quantum hardware to improve ML algorithms, even if practical execution is currently still mostly limited to simulators instead of real hardware. This emergent field of Quantum ML leads to the subject of our **second research question**.

Taken together, the present thesis is designed to address the challenges outlined in the context of typical ML business applications like customer churn prediction [81], customer segmentation [5], credit scoring [27], or product life-cycle management [104]. Firstly, regarding the problem formulation challenge, a mapping of business problems, evaluation measures, and commonly applied ML methods is established through a structured review of undergraduate student projects carried out with corporate partners in the Data Science Lab course at Vienna University of Economics and Business.

Research Question 1: *Among typical medium-scale ML projects previously conducted in cooperation with corporate partners in the Data Science Lab course at WU Vienna, which classical ML methods have proven to be successful for common business use cases?*

Secondly, the thesis aims to shed light on the applicability of QML methods, on the one hand by introducing the theoretical foundations of available techniques, and on the other hand by taking a practice-oriented perspective with a review of existing QML literature for business applications as well as an implementation of proxy cases with publicly available tools and datasets for demonstration purposes.

Research Question 2: *Given the current state of hardware, tools, and algorithms, to which extent could the application of Quantum ML methods to common business use cases be advantageous?*

The thesis is structured as follows: Section 2 describes the methodology applied to the research questions in more detail. Next, Section 3 presents the results of reviewing the Data Science Lab student projects. Section 4 lays out definitions of basic principles of quantum computing, and then proceeds by introducing QML, the variational workflow of its near-term techniques, and its most prominent challenges. This is followed by the review of literature on QML in business scenarios in Section 5. The results of the practical QML implementation work with publicly available datasets are reported in Section 6, which also includes a reflection on the personal insights and experiences obtained in this respect. Section 7 concludes the thesis.

2 Methodology

This section outlines the methodology employed in the research process, detailing the techniques and tools used to answer the research questions.

2.1 Review Process for Student Projects

Regarding Research Question 1, a systematic review and categorization of Lab projects from the undergraduate specialization in Data Science was conducted in order to provide a synthesized mapping of ML tasks and methods in business applications, with a time-frame ranging from the first run of the course in summer-semester 2017 up to its most recent iteration in winter semester 2023/24. In general, the review is based on the final reports of student groups submitted as end product to their academic project supervisors at WU Vienna. Due to the confidentiality of some of the topics and data contributed by the partner firms involved in the course, non-disclosure agreements between the author and corporate partners have been concluded in order to ensure sufficient and legally compliant coverage of projects in the performed review. Data collection was partly limited to abridged versions of student reports which only illustrated the analysis approach, instead of the actual results when applied to the (confidential) data. Similarly, the data collected for this review focus on the overall workflows from problem definition to model training and results, as developed and reported by the student groups. In addition, corporate project partners are not mentioned directly, but classified by industrial branch via the NACE Code system of the European Community [31]. Among the 99 projects reviewed, 12 did not utilize any ML methods; thus, the further analysis considers only the remaining 87 ML projects.

In line with the proposal for transparent, standardized ML model documentation via detailed model cards [67], the criteria applied to the Data Science Lab projects are not only centered around algorithms and tools, as derived from Research Question 1, but also include the chosen performance metric and general properties of the utilized data. The full list of criteria is provided below:

- **NACE Code:** Identifier for classification of economic activity. Since a given company can have multiple applicable NACE codes, the author in such cases assigned the category most related to the use case worked upon in the respective project.
- **Business Use Case:** Application context, problem to be solved in the enterprise via data-driven solutions.
- **ML Paradigm:** Approach for defining the learning problem to be solved for implementing the use case. In total, 13 different tags, based on the ML and DL methods taxonomies by Emmert-Straub & Dehmer [30] and Sarker [87] occur in the review. Multiple paradigms per project are possible.
- **Methodology:** Short textual description of goals and implemented workflow.
- **Main Algorithms:** ML methods employed in the project.
- **Tools:** Programming languages, libraries, packages, other software.

- **Primary Performance Metric:** Score used as primary goal in optimization process and reporting of results.
- **Dataset Type:** Modality of model inputs, such as numerical, textual, geographical, or image-based.
- **Number of Features:** Number of (original and engineered) features in the dataset fed into the ML model.
- **Number of Samples:** Total number of data points used for training, validation and testing of the final model, for example considering application of over- or undersampling techniques.

Given the scope of these criteria, the data collected for Research Question 1 not only allow for mapping problems with ML techniques, but also for tracing conceptual aspects (ML paradigm and performance metric selection) and practical insights (commonly used ML libraries). The dataset of Lab projects compiled for this review can be found in **Appendix A**.

2.2 QML Review and Case Selection

Turning to Research Question 2, the scope of this thesis is restricted to QML methods for gate-based quantum computing [71], as opposed to adiabatic quantum computing [3]. While both paradigms are equally expressive in general (each can simulate the other with at most polynomial overhead), applications of the latter concentrate mostly on quantum annealing, which generates approximate solutions for combinatorial optimization problems, but is not universally expressive. Furthermore, business applications of quantum annealers have already been comprehensively reviewed [112]. Thus, the introduction to QML methods in Section 4 covers the most common techniques that can be implemented with existing QML frameworks for gate-based quantum computing.

The review of QML business applications in Section 5 follows a use case-centric approach: While it takes the business use cases established for Research Question 1 as a basis, it also explores related scenarios that do not appear among the reviewed student projects. The literature search was performed via Science Direct, Google Scholar, and the arXiv preprint repository. The choice to include studies from the arXiv subject categories "Quantum Physics" and "Machine Learning" has been made in accordance with the customs of the QML research community. For all channels, search terms included the combination of "quantum machine learning" with "business applications", "industry applications", business functions like "finance" or "logistics", and the names of the business use cases. Manual screening of search results encompassed the inclusion criteria "business context of learning task", "gate-based QML methods", and "applied aspect". The latter means that purely theoretical proposals that have not been tested on real quantum hardware or simulators should at least be designed for a concrete business application to be eligible for the review.

For the practical implementation work, two publicly available, synthetic datasets from Kaggle were chosen as proxy cases to showcase QML algorithms on business use cases identified in the review for Research Question 1:

- **Churn Prediction:** This dataset emulates a typical scenario of predicting customer churn in retail banking [102].
- **Predictive Maintenance:** This is a cleaned version of the machine failure prediction dataset created for a paper on Explainable Artificial Intelligence applications [63, 64].

In running a selection of QML analyses on these data, the goal is not to optimize performance, but to illustrate the workflow in practical settings. Moreover, the cases are implemented using different Python-based programming frameworks: Firstly, the churn prediction case illustrates the usage of PennyLane [12], a toolkit specifically built to enable automatic differentiation such that users do not need to explicitly define how the gradients of their QML models are computed. This enables flexible integration with established classical ML frameworks like TensorFlow [1] or PyTorch [6] in hybrid architectures. Secondly, the predictive maintenance case uses the functionalities of the Qiskit Machine Learning library, which forms part of the general-purpose quantum software stack Qiskit. The latter recently received a major reworking with the release of its 1.0 version [45], which is used in the predictive maintenance example.

By employing both PennyLane and Qiskit Machine Learning for practical examples, the thesis aims to provide an intuition about their similarities and differences. Overall, these two frameworks arguably represent the most commonly used choices for implementing QML methods on practical applications. Both allow for running algorithms on real quantum hardware accessible via cloud, but also provide a broad range of basic and high-performance simulators. Given the limited accessibility of quantum hardware, both examples in this thesis were run on simulators. While switching to real hardware would require careful choice of transpilation options (making the high-level quantum algorithm executable on a concrete quantum device) and error mitigation measures (minimizing noise arising from physical qbit implementation on a real quantum device), the overall coding workflow apart from these additions does not change, making the qualitative insights drawn from the performed implementation process generalizable to a large extent.

3 Review of Data Science Labs

Reviewing the 87 ML projects from previous Data Science Lab courses at WU Vienna, this section will first present the results of analysing single comparison criteria, and then proceed by describing the established mapping of problems, algorithms, tools, and metrics.

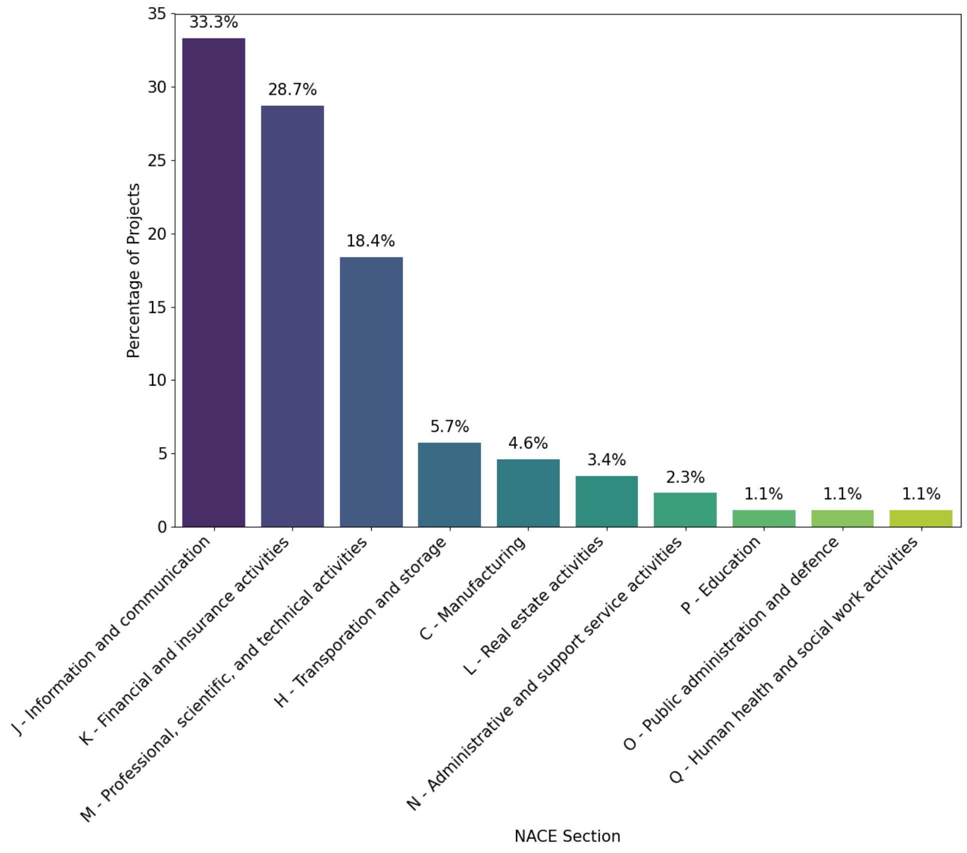

Figure 1: Percentage distribution of NACE section classification among Data Science Lab project partners

To illustrate the broad range of business sectors covered by the corporate partners that have been involved in the ML student projects, the percentage distribution of **NACE sections** these firms belong to is shown in Figure 1. The NACE section here corresponds to the first letter of the NACE code and thus represents a high-level classification. From the bar plot, it can be seen that there is a concentration on three areas: Firstly, Section J, which accounts for one third of projects, includes business activities such as news agency activities (J6391), data processing (6311), IT consulting (J6202), or software products (J5829). The second, almost as most prominent field is Section K, which mainly consists of activities like classical banking (K6419), but also financial regulation (K6611, K6411). Section M, which is placed on third position with a share of 18.4%, encompasses activities of accounting, auditing, and tax consultancy (M6920), as well as various kinds of consultancy in management and technical issues (M7022, M7112, M7219, M7311). While the dominance of these three sections aligns with the high demand for data-driven analyses especially in the IT and financial sectors, the presence of several other sectors among project partners ensures that use cases from a sufficiently broad range of business contexts are included in the present review.

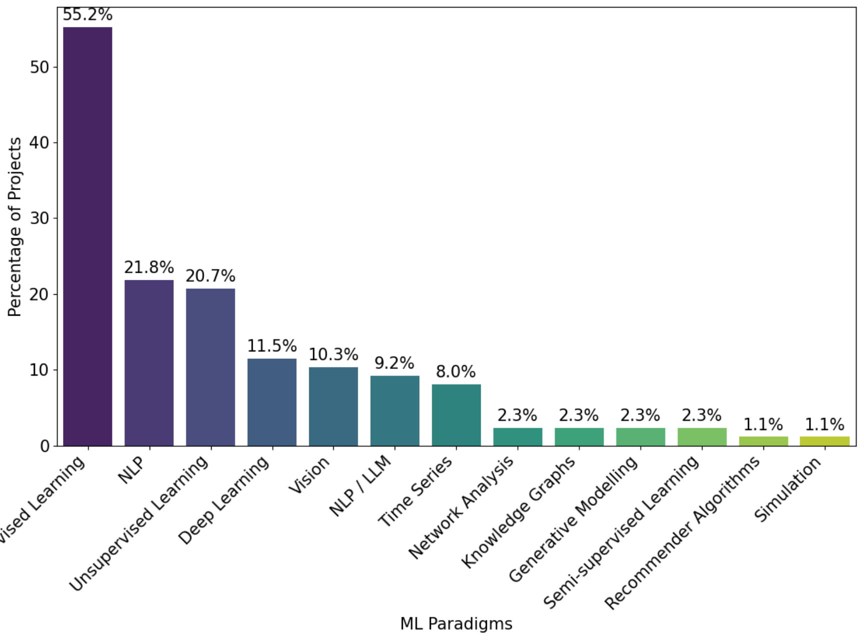

Figure 2: Broad ML paradigm usage by percentage of Lab projects

Turning to the characteristics of the projects themselves, we can examine the two main aspects of the problem formulation stage in the Data Science process: On the one hand, this includes the framing of business problems in terms of ML paradigms, taking into account the modality of available data. On the other hand, this crucial step encompasses the choice of metric for measuring and optimizing the quality of the results.

Figure 2 displays the share of each **ML paradigm** among the 87 projects. Note that a single project can be tagged with multiple paradigms - for example, image classification would belong to the Supervised Learning and Vision paradigms. Conversely, Supervised Learning as the only tag implies standard numerical data inputs, while specific fields like NLP or Recommender Systems can include domain-specific methods outside the typical Supervised versus Unsupervised ML distinction. Overall, the distribution demonstrates the high prevalence of the Supervised ML paradigm, i.e., settings where labelled data are available, with more than half of the cases falling in this category. By contrast, Unsupervised ML, which encompasses descriptive and generative models on unlabelled data, is only used in approximately 20% of projects. The low share of 11.5% for Deep Learning might seem surprising, given the current hype around these methods. However, it has to be kept in mind that their surge in popularity is a relatively recent phenomenon, and the requirements of student projects might favor traditional ML algorithms due to lower implementation and training efforts as well as explainability considerations. Furthermore, reliance on traditional, robust ML methods was explicitly encouraged by academic supervisors in case of small- to medium-size datasets. Instead of training own DL models, the more recent projects increasingly adopted the application of pre-trained Large Language Models (LLMs) for tasks performed on text data. Indeed, the ability to deal with the latter is underscored as an important skill in corporate Data Science environments, given that almost one third of projects employed either LLMs or traditional NLP methods. The Vision and Time Series paradigms also occurred more frequently with 10.3% and 8% coverage, while the remaining six tags were are only associated with one or two projects each.

Overall, it can be argued that the distribution of ML paradigms is well-aligned with the **distribution of data types** among the datasets utilized in the projects, which is as follows:

- **Numerical data** are the most prominent category with a share of 49.4%. Typically, these data are already collected / compiled by project partners and made available as tabular samples.

- **Text data** were used in 35.6% of projects, many of which used web scraping or extraction from provided documents to create the datasets.
- **Images**, for example from satellites or for text extraction, were utilized in 9.2% of cases.
- **Other data types** in 5.7% of projects included geo-data, graph-shaped data, and linked data.

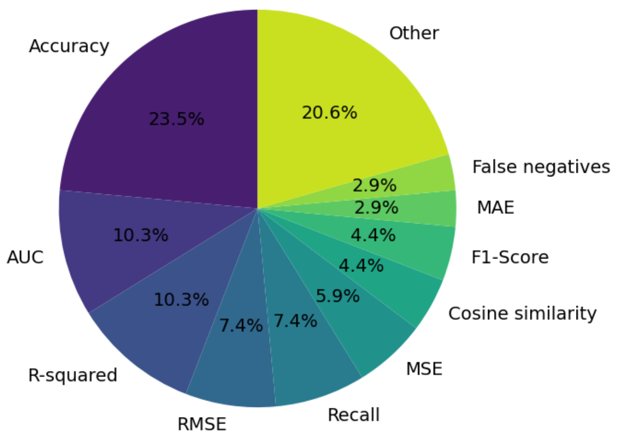

Figure 3: Distribution of main performance metrics among Lab projects

The second major aspect of problem formulation is the choice of appropriate **performance metrics**. While ML projects can of course be directed at optimizing multiple metrics simultaneously, the focus of the review is on the primary criterion per project, i.e., the criterion selected for optimization in the training procedure. The distribution of metrics shown in Figure 3 clearly underscores the importance of deriving the evaluation mode from business needs and being aware of the consequences for model results. In total, 24 different options occur among the Lab projects, covering not only metrics for supervised ML, but also for unsupervised models, like cosine similarity, perplexity, or BERT score. For a comprehensive theoretical account of these and other ML performance metrics, we refer the reader to Chapter 5 of the Probabilistic Machine Learning book by Murphy [48]. Regarding classification, accuracy is often conceived of as default choice for performance measurement, and it is indeed confirmed as the most frequently used option among Data Science Lab projects, as seen from Figure 3. At the same time, however, the metric distribution also demonstrates that many classification projects adopted alternative measures like Area-Under-the-Curve (AUC), Recall, F1-Score, or number of False Negatives. A similar picture is obtained for regression problems, where R-squared is most frequently used, followed by Root Mean Squared Error (RMSE), Mean Squared Error (MSE), and Mean Absolute Error (MAE). A notable peculiarity is the usage of evaluation measures outside the statistical realm, such as response time (model inference) or holistic judgment of outputs, for some text generation tasks among the Lab projects. This once more underscores the significance of adopting a broader perspective when evaluating ML models.

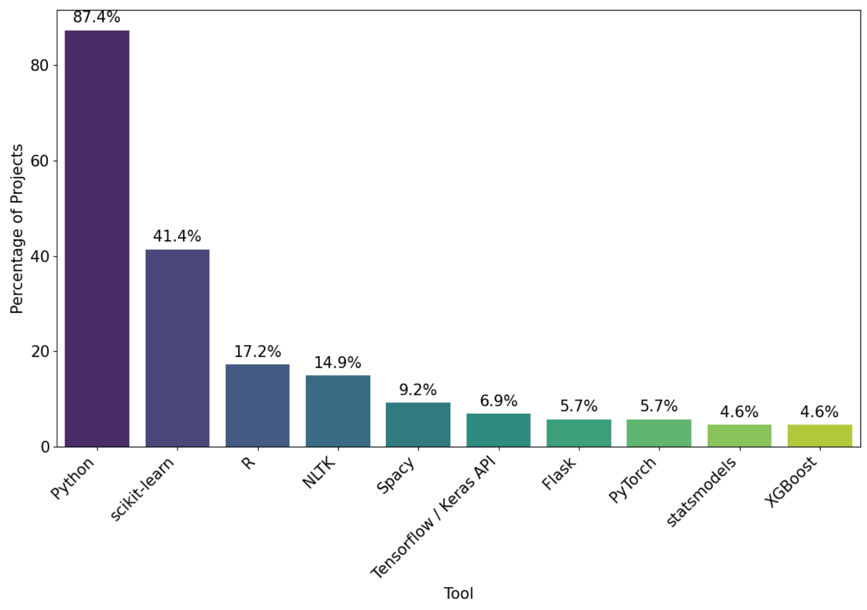

Figure 4: Ranking of top 10 tools by share of utilization among Lab projects

As a practical perspective on project implementation, Figure 4 presents the **10 most frequently used tools**. The ranking clearly reveals Python [83] as the overall more significant programming language for ML projects, with a five times higher coverage compared to R [84], which is more rooted in traditional statistics. However, it is important to note that some Lab projects used both languages, showing that the functionalities offered by Python and R can complement each other. Among dedicated ML libraries, sklearn [78] emerges as the most popular option with a share of more than 40%. Further prominent tools for running ML algorithms are the DL frameworks TensorFlow [1] and PyTorch [6], as well as the XGBoost library [21] for gradient boosting algorithms. Reflecting the distribution of dataset types, the NLTK [14] and spaCy [40] libraries for wrangling and analysing text data with NLP methods are found on places 4 and 5 in the ranking with shares of approximately 15% and 9%, respectively. Lastly, the top 10 tools also contain the Web framework Flask [75] for building applications such as interactive dashboards, as well as the statsmodels library [92] for statistical modelling. In summary, this ranking contains essential ingredients for building a full workflow from preprocessing the data over running ML algorithms to deploying an end product for the business customer.

Building on the insights obtained from the individual criteria outlined so far, the systematic approach of grouping projects and their characteristics by **business use case** represents the most essential result of the review of Data Science Labs. In total, 21 use case categories with varying degrees of specificity have been identified - for example, *Customer Segmentation* is a very specific task, while *Other Marketing Analytics* as a more general category contains multiple tasks like website analytics or generation of targeted advertisements. The results of mapping each of these use cases to their most frequently used ML paradigm, as well as their top 3 algorithms, libraries, and evaluation metrics are presented in Table 1. The overall ordering of use cases in the table is from most frequent to least frequent category. Moreover, the items in the columns for algorithms, libraries, and metrics are ordered by their frequency among projects belonging to the given use case.

Observing the results, it is especially interesting to see *Data Collection and Enrichment* on the top position in the frequency ranking. In this use case, mostly text-based (traditional NLP or LLM) methods were used to leverage diverse data sources such as websites, emails, publicly available documents, and company-internal reports. We can thus confirm that ML methods do not only play their natural role in the modeling step of the Data Science process, but also support earlier tasks of collecting the data and improving their quality. At the same time, they can also help companies in compliance with privacy regulations on data usage, as seen from the synthetization of images in the *Data Governance* use case.

Table 1: Mapping of use cases to main ML paradigm, algorithms, libraries, and evaluation metrics.

Business Use Case	**Main ML Paradigm**	**Main Algorithms**	**Main Libraries**	**Main Evaluation Metrics**
Data Collection and Enrichment	NLP	GPT, Naive Bayes, LDA	sklearn, NLTK, SpaCy	Accuracy, BERT Score, Cosine Similarity
Credit Risk Management	Supervised ML	Decision Tree, Random Forest, Logistic Regression	sklearn, PySpark, imbalanced-learn	AUC, Accuracy, RMSE
Customer Segmentation	Unsupervised ML	kMeans Clustering, DBSCAN, CLARA	stats, dtplyr, sklearn	Accuracy, AUC
Valuation	NLP	kMeans and hierarchical Clustering, MDS	sklearn, NLTK, SpaCy	Cosine Similarity
Churn Prediction	Supervised ML	Decision Tree, Random Forest, Logistic Regression	sklearn, imbalanced-learn, caret	Recall, False Negatives, F1-Score
Logistics Automation and Analytics	Supervised ML	LSTM, Conv1D, MLP	sklearn, Tesseract, TensorFlow	Accuracy, R-squared
Demand and Price Prediction	Supervised ML	Random Forest, SARIMAX, Gradient Boosting	sklearn, statsmodels, XGBoost	MSE, RMSE, MAE
Other Marketing Analytics	Supervised ML	LSTM, GPT, Deepmatcher	PyTorch, TensorFlow, GPT3	Recall, F1-Score, R-squared

Business Use Case	**Main ML Paradigm**	**Main Algorithms**	**Main Libraries**	**Main Evaluation Metrics**
Social Media Analytics	NLP	SVM, Linear Regression, Naive Bayes	NLTK, SpaCy, sklearn	Accuracy, AIC, R-squared
Infrastructure Planning	Time Series	SARIMA, U-Net CNN, LSTM	statsmodels, sklearn, Tensor-Flow	Precision, R-squared, MAPE
Chatbot	NLP	GPT, LSTM	GPT4, Pandas Langchain Agent, Tensor-Flow	Accuracy, Response Time, F1-Score
Predictive Maintenance	Supervised ML	Linear Regression, Logistic Regression	mlr, caret, SAP Analytics Cloud	MSE, RMSE
IoT Application	Supervised ML	Gradient Boosting, Random Forest, Tabular VAE	mlr, xgboost, Tensor-Flow lite	TPR, FPR, ROC Curve
Data Governance	Vision	GAN, CNN, Gaussian blurring	OpenCV, Tensor-Flow, scikit-image	RMSE, Correlation Matrix, FNR
Applied Environmental Research	Supervised ML	LSTM, Linear Regression	sklearn, stats-models, PySpark	Accuracy, R-squared
Medical	Supervised ML	Linear Regression, Decision Tree, Random Forest	SpaCy, NLTK, sklearn	Accuracy, AUC, R-squared

Business Use Case	**Main ML Paradigm**	**Main Algorithms**	**Main Libraries**	**Main Evaluation Metrics**
Automated Reporting	NLP	LSTM, BERT, GPT	NLTK, Tensor-Flow, PyTorch	Accuracy
Recommender System	Unsuper-vised ML	kMeans Clustering, Alternating Least-Squares, Bayesian Personalized Ranking	sklearn, implicit, LightFM	AUC
HR Management	Supervised ML	Random Forest, Naive Bayes, Gradient Boosting	imbalanced-learn, sklearn, XGBoost	AUC
Process Analytics	Supervised ML	Decision Tree, Jenks-Breaks Algorithm	sklearn, jenkspy	Accuracy
UX Improvement	NLP	Pre-trained topic modelling classifier	gensim	Recall

While the above mapping does not provide an exhaustive list of ideal problem formulations for the listed use cases, it can serve as an instructive demonstration of the broad range of tasks where the utilization of ML algorithms can help to deliver business value. Together with the full dataset of ML projects among previous Data Science Labs, it also constitutes a knowledge repository for future iterations of such projects. Finally, the collection of use cases will be revisited in Section 5 where the application of Quantum Machine Learning (QML) methods to business problems will be reviewed.

4 Fundamentals and Challenges of QML

This section first introduces a selection of necessary concepts from quantum information theory and gate-based quantum computing which contains the essential building blocks of current QML methods. Next, the QML research field is introduced, and the workflow of variational QML algorithms is examined in a step-by-step manner. The section closes with an overview on challenges that these methods and applied QML research in general are currently facing.

4.1 Basic Principles of Quantum Computing

In order to understand how QML methods on gate-based quantum computers work and what they can or cannot achieve, it is first of all necessary to establish a solid comprehension of some basic quantum computing principles. Thus, this subsection provides an introduction for the reader not familiar with these concepts, giving preference to intuitiveness instead of full mathematical rigor. The main sources for this part are the standard QML textbook by Schuld & Petruccione [91] and the QML introduction for non-practitioners in Byrne et al. [17], which may be consulted by the interested reader for further reading on conceptual QML basics. Regarding mathematical background, the textbook by Axler [8] offers an accessible account of the underlying Linear Algebra concepts, while Nielsen and Chuang [71] provide an advanced formal introduction to the quantum-mechanical framework.

The first question that we naturally ask when moving from classical to quantum computing is: how do we represent information? The answer is that the binary representation given by the classical bit still plays a role, but forms part of a more general and expressive definition. This can be seen from the definition of the quantum bit (**qbit**), which is given below in the so-called Dirac notation for vectors (also called bra-ket notation) that is commonly used in quantum computing:

$$|\varphi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle , \tag{1}$$

$$\text{where:} \quad |0\rangle = \begin{pmatrix}1\\0\end{pmatrix}, \quad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}, \tag{2}$$

$$|\alpha_0|^2 + |\alpha_1|^2 = 1, \tag{3}$$

$$\alpha_0, \alpha_1 \in \mathbb{C}. \tag{4}$$

This is the general definition for a qbit in an arbitrary quantum state $|\varphi\rangle$, which is a column vector, spoken as "ket phi". More precisely, we can see that the state is a linear combination of the states $|0\rangle$ and $|1\rangle$, which (2) identifies as classical bits. From a linear algebra perspective, $|0\rangle$ and $|1\rangle$ form a basis for the two-dimensional vector space, also named the *computational basis*. The coefficients α_0 and α_1 are called the *amplitudes* of the corresponding basis states. Note that by condition (4), amplitudes α_j in general are complex numbers that can be written as:

$$\begin{aligned} \alpha_j &= x + iy && (5) \\ &= r * (cos\theta + i * sin\theta) && (6) \end{aligned}$$

where $r, x, y \in \mathbb{R}$, $\theta \in [0, 2\pi)$, and $i = \sqrt{-1}$. The two definitions above are equivalent - the two dimensions of real and imaginary parts in complex numbers can be expressed in the *canonical form* (5) via two coordinates x, y or in the *polar form* (6) via radius r and angle θ. The relation between parameters of both forms is illustrated in Figure 5 below.

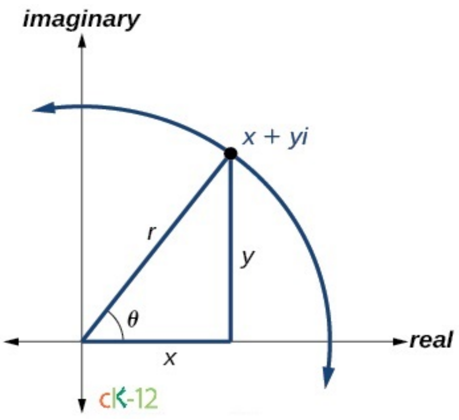

Figure 5: Illustration of the 2D complex plane [33]

Of course, many practical cases confine themselves to purely real amplitudes, but in principle, the framework of quantum information theory works with the more expressive complex numbers. In any case, the qbit definition contains a restriction on the amplitudes in (3), stating that a valid quantum state must be normed, i.e., of unit length. The absolute value of a complex amplitude α_j here corresponds to the radius r in the polar form:

$$|\alpha_j| = r = \sqrt{x^2 + y^2} \qquad (7)$$

The reason for imposing the normed length condition on qbits lies in its interpretation: An arbitrary qbit $|\varphi\rangle$ as in (1) is said to be in **superposition** of its two basis states, each of which is associated with a certain probability given by the squared absolute amplitude $|\alpha_j|^2$. Thus, these probabilities naturally need to add up to 1 in total. As a consequence of the superposition principle in the qbit definition, we only need n qbits to encode 2^n classical states. For example, if we set $n = 2$ bits, we obtain the four computational basis states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. To represent a superposition of these four states, we then only need two qbits. In the simplest case of equal probability for each basis state, the overall state is given by:

$$|\phi\rangle = \frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right) \tag{8}$$

Thus, for $n = 2$, we can work with $2^2 = 4$ classical states at the same time. In other words, any operator acting on a state with n qbits manipulates the amplitudes of an exponential amount of 2^n basis states - this capability is known as **quantum parallelism** and represents one ingredient how quantum algorithms can achieve speedups compared to classical algorithms.

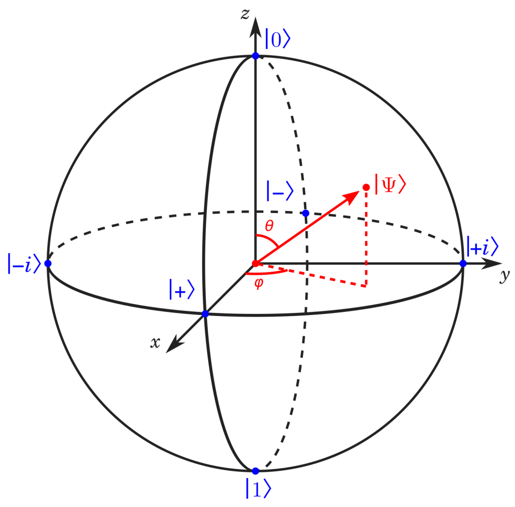

Figure 6: Single qbit quantum state as a point on the Bloch Sphere [70]

The meaning of superposition can also be seen from the well-known visualization of a single qbit state as a point on the Bloch sphere, as depicted in Figure 6. While the computational basis states $|0\rangle$ and $|1\rangle$ are located at north- and south-pole, the qbit depicted in red lies somewhere in between on the surface of the sphere, on a point fixed by two angles in the interval

$[0, 2\pi)$. Since the variations of these angles can be arbitrarily small, there are uncountably many possible single qbit states! Moreover, the Bloch sphere shows that using classical bits $|0\rangle$ and $|1\rangle$ as our basis for representing the qbit state $|\varphi\rangle$ in equation (1) is not the only option - any other set of diametrically opposed points, such as $\{|+\rangle, |-\rangle\}$ or $\{|i\rangle, |-i\rangle\}$, could equally serve as basis. Similarly to the change of coordinate representation when performing a basis change in linear algebra, the amplitudes of a quantum state would then change as well. However, the computational basis is the default choice for a good reason, since practically relevant quantum algorithms typically include classical post-processing and thus need classical bits as outcomes of the quantum computation.

Now how can we perform computations on qbits? In the gate-based quantum computing paradigm, every operation is represented by a *quantum gate*, while a combination of several gates on one or more qbits forms a *quantum circuit*, analogous to the Boolean logic circuit in classical computing. In quantum computing, all gates share the property that they represent **unitary operators**, i.e., a special class of matrices. Unitary matrices exhibit two essential characteristics: On the one hand, their inverse always exists, ensuring that in contrast to classical Boolean circuits, all operations are *reversible*. On the other hand, unitary operators are *length-preserving*, such that the application of gates always yields a valid quantum state conforming with the normed length condition as a result.

For the case of **single-qbit gates**, we can take a look at Figure 6 again and think of operators as rotations around one or more of the three axes: By the properties of unitary matrices, we can always go back and forth between any pair of points on the Bloch sphere, and we can be sure that we will stay on the surface. The operators that realize these rotations around the axes are called the *Pauli matrices*. In their standard form written down below, they perform a rotation of 180 degrees:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{9}$$

In the computational basis, X can be thought of as a classical bit flip, while Z performs a phase flip by changing the sign of the amplitude for $|1\rangle$. However, the Pauli matrices can also be parameterized by a given angle $\theta \in [0, 2\pi)$. The resulting operators $R_X(\theta), R_Y(\theta), R_Z(\theta)$ form a set of universal operators that can realize any possible rotation on the Bloch sphere. Nevertheless, there is one more single-qbit gate that should not be missed due to its common occurrence in quantum algorithms.

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{10}$$

The Hadamard gate defined in (10) enables the preparation of qbits in superposition from a register of qbits initialized as classical bits $|0\rangle$ or $|1\rangle$, as shown below.

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle \tag{11}$$
$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle \tag{12}$$

In both cases, the computational basis states $|0\rangle$ and $|1\rangle$ are associated with a probability (squared absolute amplitude) of $\frac{1}{2}$, which means that the qbits are in equal superposition. Since the resulting states $|+\rangle$ and $|-\rangle$ are also diametrically opposed points on the Bloch sphere, this procedure is an example for a change of basis, and these states are sometimes called the *Hadamard basis.*

With the extension to **multi-qbit gates** acting on a register of qbits, it becomes desirable to express more complex patterns. Most importantly, this is enabled by *controlled operators* that resemble an *if-else* logic: An operation is only performed on the *target qbit* dependent on the value of one or more *control qbit(s).* For the purpose of this introduction, we exemplify this with the *Controlled NOT (CNOT)* gate. If the control qbit is in state $|1\rangle$, this gate executes a bit flip on the target qbit, and if not, the identity operation is applied. Since it acts on two qbits, $CNOT$ is represented by a 4 x 4 matrix:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{13}$$

Before we can build a small example of a full quantum circuit, we still have to clarify how we can **obtain information** about the quantum states that result from the application of a sequence of gates. By the laws of quantum mechanics, we do not have a way of directly observing a quantum state as defined in (1) - we can only read out classical information. Thus, the act of observation in a sense "destroys" information, and in contrast to application of gates, it is not reversible. Overall, there are two options to obtain information about a state:

- **Measurement:** Executing the circuit many times to count the occurrences of each basis state. This corresponds to a sampling procedure and yields an estimated probability distribution.
- **Expectation value:** Estimated by the average outcome of many measurements ("shots"). This results in a single numerical value.

The number of required shots can be derived from the desired error margin for the result, considering the sampling nature of the process and, if applicable, the probability of readout errors on a certain real quantum device. For both measurement and expectation values, we also need to choose an **observable** which defines the aspect of the system and its associated possible outcomes that should be measured. In physics, this could for example be the position of a particle, defined by a grid of all possible positions. In quantum computing, the most natural choice is the computational basis, i.e., all possible combinations of classical bits $|0\rangle$ and $|1\rangle$ in an n qbit system. Any valid choice of observable must adhere to a set of requirements: Firstly, measurement outcomes observed in the real world are always real numbers. Therefore, an observable should also provide us with only real numbers, as opposed to complex numbers with imaginary part. Secondly, we need to be able to precisely distinguish between all possible outcomes, meaning that the outcomes must form a complete set of mutually orthogonal basis states.

In mathematical terms, these requirements are fulfilled by any *Hermitian* matrix. Most importantly, measurement does not mean to *apply* such a matrix to a state by computing the matrix-vector product. Instead, measurement constitutes a *projection* of the state vector onto the eigenvectors of the observable, with each eigenvalue acting as indicator for a particular eigenvector. Since a Hermitian matrix has only real eigenvalues, we have a guarantee that the correspondence between eigenvalues and observed measurement results works as required. Moreover, Hermitian matrices always have an orthonormal basis of eigenvectors (mutually orthogonal, unit length) and can be represented as the sum of projections onto these eigenvectors (*spectral theorem*). The unit length condition is again important for quantum-mechanical interpretation - after the projection, the probability to be in the obtained state (eigenvector) is 1. Overall, the spectral theorem shows that Hermitian matrices can be understood as a combination of directions (eigenvectors) that span the whole space (basis) and are weighted by their importance (eigenvalues). Thus, our second requirement for observables is fulfilled as well.

From a practical point of view, we have already mentioned that in quantum computing, we are in most cases interested in retrieving classical bits from measurement. This is why most quantum devices physically implement only a single observable: The *Pauli Z* matrix in (9) is a Hermitian matrix with eigenvectors $|0\rangle$ and $|1\rangle$, associated with eigenvalues 1 and -1, respectively. Measurement of a general qbit $|\varphi\rangle$ as in definition (1) with the Z observable then yields the eigenvalue 1 with probability $|\alpha_0|^2$ and eigenvalue -1 with probability $|\alpha_1|^2$. To use a different observable, ancillary computations like decomposing the observable or transforming it with unitary operators typically enable a formulation in terms of the physically supported Z measurement. The same holds for the computation of *expectation values*. When choosing the Z observable, the expectation value of $|\varphi\rangle$ reduces to subtracting the probability to obtain $|1\rangle$ from the probability for $|0\rangle$:

$$\langle Z\rangle_{|\varphi\rangle} = \langle\varphi|\, Z\, |\varphi\rangle = |\alpha_0|^2 * 1 + |\alpha_1|^2 * (-1) = p(|0\rangle) - p(|1\rangle) \tag{14}$$

The expression $\langle\varphi|\, Z\, |\varphi\rangle$ above can be interpreted as measuring the degree to which the application of Z on $|\varphi\rangle$ aligns with the state $|\varphi\rangle$ itself. Thus, the projection is an estimation for the average of many measurements.

Now we have covered all the components necessary to build a quantum circuit in a small example on two qbits. At the same time, we will use the example circuit to discuss **entanglement**, a phenomenon that is unique to quantum mechanics.

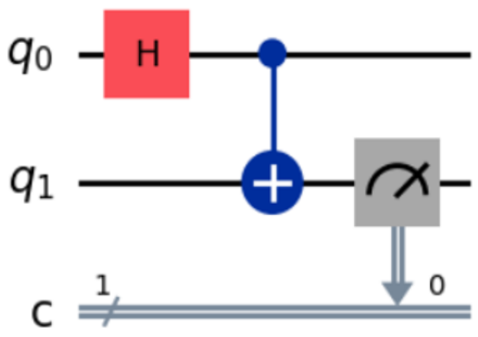

Figure 7: Quantum circuit for generating entangled Bell state $|\Phi^+\rangle$, created by the author with Qiskit 1.0

Inspecting the quantum circuit displayed in Figure 7, we can see that it consists of a quantum register containing two qbits q_0 and q_1, as well as a classical register C containing a single bit. Thus, it has a *circuit width* of 2. The *circuit depth* is 2 as well, since we have a total of 2 layers of operations that cannot be parallelized. In the beginning, each qbit is in the $|0\rangle$ ground state, which is written as $|00\rangle$ for the overall two-qbit system. We will now follow the series of gates and measurement from left to right. Firstly, we have a Hadamard gate on q_0, the effect of which we have already encountered in equation (11). This leads to following overall state:

$$\frac{1}{\sqrt{2}}\left(|00\rangle + |10\rangle\right) \tag{15}$$

The second gate in the circuit is a *CNOT* gate with control qbit q_0 and target qbit q_1. Since we are using computational basis states, we can think of it as a conditional bit flip: The $|00\rangle$ basis state in equation (15) does not change since it contains a 0 for q_0, but for the second basis state $|10\rangle$, the bitflip on q_1 is executed. We are now left with the following two-qbit state:

$$\frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) = |\Phi^+\rangle \tag{16}$$

This state is denoted by $|\Phi^+\rangle$ and belongs to the set of four *Bell states*, which are well-known for being *maximally entangled*. To understand what this means, we have to consider the measurement at the end of our circuit in Figure 7. We only measure q_1 and save the measurement outcome (eigenvalue indicating either $|0\rangle$ or $|1\rangle$) in a classical bit. By the entangled nature of the $|\Phi^+\rangle$ state, we then immediately know the value of q_0 without the need to perform a further measurement: If we receive $|0\rangle$ for q_1, it is clear that we can only be in the two-qbit basis state $|00\rangle$, while for $|1\rangle$ as measurement result, we necessarily are in basis state $|11\rangle$. Since each of these cases occurs with probability of $\frac{1}{2}$, we speak of a *maximally entangled* state. The effect of entanglement is in principle independent of the distance between the particles belonging to the entangled state: Experiments conducted mainly with photons by Nobel laureate Anton Zeilinger and colleagues have confirmed the non-locality of the phenomenon which cannot be explained with classical physics [60]. As a form of powerful non-classical correlation, we will see that entangled states also play a role in QML algorithms, which are introduced in the next subsections.

4.2 Overview of QML Methods

Quantum Machine Learning (QML) is a comparatively young research field that saw its first seminal publications around 2014 [91]. Since then, it has developed a steeply growing research activity, with the number of relevant publications per year almost reaching the 1400 mark in 2022 [103]. When asking for a **definition** of its scope, it seems natural to describe it as the unification of the capabilities provided by quantum computing and ML algorithms. However, Schuld & Petruccione [91] show that there are in fact four different intersections between these two domains, which are displayed in Figure 8 below.

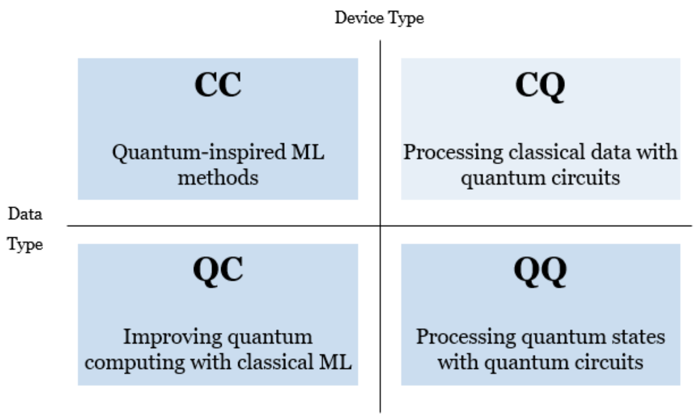
Device Type
Data Type
CC
Quantum-inspired ML methods
CQ
Processing classical data with quantum circuits
QC
Improving quantum computing with classical ML
QQ
Processing quantum states with quantum circuits

Figure 8: Four intersections of quantum computing and machine learning by data and device type, adapted from Figure 1.1 in Schuld & Petruccione [91]

Firstly, the *CC* intersection denoting classical ML executed on conventional hardware shows that quantum mechanics can serve as an inspiration, for example by using tensor networks as a data structure in DL algorithms [91]. Furthermore, some quantum algorithms that are out of reach for realization on current quantum computers have been "dequantized" by utilizing an equivalent classical routine with similar speed [105]. Next, the *QC* intersection aims to leverage classical ML to improve the performance of quantum hardware. In particular, the transpilation process [46] that maps a quantum circuit to the physical qbits and native gates of a specific quantum device can benefit from applying deep reinforcement learning methods instead of traditional heuristic methods [50]. Turning to the case of processing quantum data on quantum devices, this *QQ* intersection seems like a natural fit: For complex learning tasks like explaining the dynamics in experimental data from physics and chemistry, utilizing quantum circuits can avoid bottlenecks of data representations and potentially better detect patterns that are characteristic for quantum systems. However, this approach is of course restricted to applications in the natural sciences [13].

Thus, what is typically meant when referring to QML algorithms is the *CQ* intersection, where classical data are processed with quantum circuits. More precisely, this setting always includes both classical and quantum hardware, as well as an interface for data exchange between the two components. Therefore, we can define these QML methods as *hybrid quantum-classical workflows for learning from classical data*, while being aware that quantum computing and ML share a set of further intersections [91]. As a further side note, we have to keep in mind that the above systematization only includes ML relations of gate-based quantum computing - the usage of quantum annealers for optimization procedures in DL algorithms is sometimes referred to as part of QML, but conceptually rather belongs to combinatorial optimization [13].

With our general QML definition being established, we also have to mention the **two branches** that have developed in the research field. The branch that is more rooted in the quantum computing tradition is called **Fault-tolerant QML**. As indicated by its name, this paradigm deals with algorithms designed for ideal devices that can perform many computations on a large number of qbits, which are still far from being realized. Thus, this branch is directed at theoretically proving advantages over classical computing in terms of computational complexity [91]. Most fault-tolerant QML proposals purely focus on a linear algebra perspective: By leveraging quantum routines for Fourier transforms, finding eigenvalues and eigenvectors, and solving linear systems of equations, for example, their aim is to speed up well-known ML algorithms that rely on these linear algebraic operations, such as PCA or SVM. In addition, many speedups in this category assume availability of data via Quantum Random Access Memory (QRAM), which would allow to reliably store quantum states and access multiple memory locations simultaneously via superposition. However, this is still a purely theoretical construct without existing realization [105]. By contrast, **Variational QML** is a paradigm that aims to make use of the resources that are available today. While NISQ hardware offers only few qbits and imposes restrictions on gate fidelity (probability of error-free execution) and decoherence time (time until qbit states get corrupted and information is lost), execution of larger circuits can quickly become intractable on simulators due to the exponential growth of the state space with increasing number of qbits [82]. Thus, variational algorithms have to be rather compact and use conventional classical methods for parts of the workflow to comply with these limitations [105]. However, this branch also differs from fault-tolerant QML by adopting an explicit ML mindset and really thinking of quantum circuits as models. In fact, variational quantum algorithms represent a new ML model class that can be analysed with a rich set of tools from both linear algebra (kernel theory) and functional analysis (universal approximation properties, Fourier series). In evaluating proposed methods, variational QML predominantly relies on empirical benchmarking, even though we will see in subsection 4.4 that this poses several significant challenges [91].

4.3 The Variational QML Workflow

The generic workflow of variational QML models follows a hybrid approach where parameterized circuits are executed either on a quantum device or a simulator, while the cost function depending on the chosen parameters is always evaluated and optimized on a classical device. We will now introduce this process, which is depicted in Figure 9, by examining its parts in a step-by-step manner.

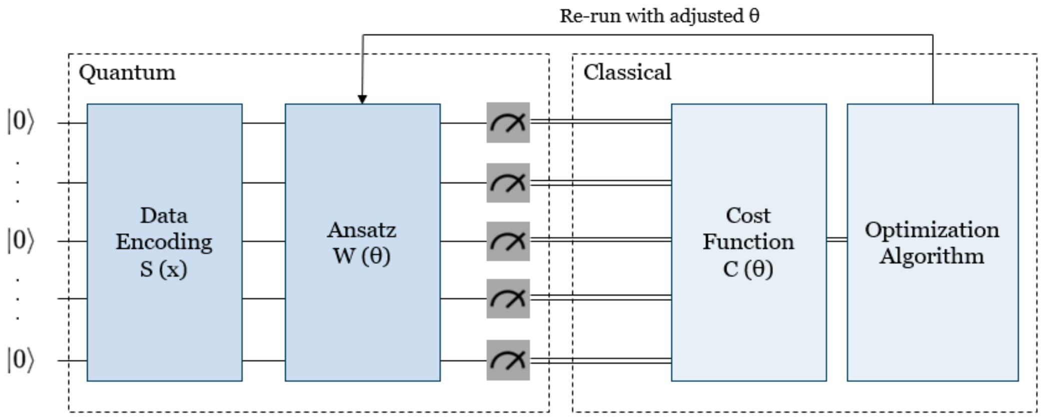

Figure 9: Hybrid workflow of variational QML models, adapted from IBM Quantum Learning [53] with notation of Schuld & Petruccione [91]

4.3.1 Data Encoding

Firstly, the choice of **data encoding** defines how each D-dimensional data point $\mathbf{x}_i$ of a classical dataset is represented on n qbits, either as a separate quantum state $|\varphi_i\rangle$ or in a superposition $|\varphi\rangle$ with all other data points. Technically, this is realized as a unitary quantum circuit $S(\mathbf{x})$ that prepares this state from a $|0\rangle$-initialized n qbit register. In the quantum computing literature, other equivalent terms for this procedure include state preparation [106], reference preparation [53], data loading [82], data embedding [17] and feature embedding [89]. As in the later steps of the variational workflow, the reason for this lack of coherent terminology is that each part of the process can be viewed from the perspective of arbitrary quantum circuits, optimization, or ML theory. In addition, data encoding has been conceived of as a set of patterns, emphasizing the software engineering aspect [106].

In any case, the choice of data encoding method is not straightforward, especially in view of the limited number of qbits in the NISQ era. Thus, when working with high-dimensional datasets, a common remedy is to first apply dimensionality reduction techniques and encode only the reduced data as quantum states [17, 22, 61]. Furthermore, it should be noted that the data encoding step of the workflow naturally prohibits logarithmic runtime, since every data point has to be read once, leading to at least linear runtime [91]. Nevertheless, different methods for encoding classical data into quantum states are characterised by trade-offs between factors like number of required qbits, circuit depth, and expressibility (size and complexity of encoding space), which may be more essential than runtime for ML model evaluation.

In this introduction, we briefly examine the three most common types of data encoding. Firstly, *basis encoding* [106] represents classical data in the computational basis states of an n-qbit state. This is done by transforming each feature d of a sample $\mathbf{x}_i$ into a binary string representation $|b_d^1 \dots b_d^n\rangle$ with $b_d^m \in \{0,1\}$ for $m \in \{1,\dots,n\}$. Each binary string b_d^m corresponds to a specific basis state in the n-qbit system. When considering a superposition of these basis states, we get the following n-qbit quantum state for a data point with D dimensions:

$$|\varphi_{\boldsymbol{x}_i}\rangle = \frac{1}{\sqrt{D}} \sum_{d=1}^{D} |b_d^1 \dots b_d^n\rangle \tag{17}$$

For practical illustration, we can apply definition (17) to a small example: Let $\boldsymbol{x}_1$ be a data point with three features that take values 4, 7, and 15. The binary representations of these integers are 100, 111, and 1111. Since the longest binary string consists of four bits, we need to pad all other strings to a length of $n = 4$, leading to the following basis states:

$$b_1 = |0100\rangle, b_2 = |0111\rangle, b_3 = |1111\rangle \tag{18}$$

To obtain the basis encoding of the whole data point, we create an equal superposition of the above basis states, which is given by:

$$|\varphi_{\boldsymbol{x}_1}\rangle = \frac{1}{\sqrt{3}} (|0100\rangle + |0111\rangle + |1111\rangle) \tag{19}$$

Note that discretizing continuous features for converting them to bit-strings leads to a loss of information. Furthermore, the number of qbits required for basis encoding of a single sample is given by the largest number of bits required for the binary representation of any of its feature values. Therefore, each classical bit corresponds to a qbit, which is not very space-efficient, given that a single qbit could hold values for 2^n bits.

By contrast, *amplitude encoding* [106] offers the advantage of requiring only $\lceil log_2 (D) \rceil$ qbits to represent a D-dimensional sample, and $\lceil log_2 (N * D) \rceil$ qbits for a superposition of all samples. However, the price for this compact representation is that the circuit $S(\mathbf{x})$ then becomes exponentially deep in n. Another difference to basis encoding is that continuous data can be encoded more naturally in the amplitudes. The only pre-processing requirements are normalization of each data point such that $\sum_i |\mathbf{x}_i|^2 = 1$, and padding of dimensionality D to a power of two, such that a valid quantum state is produced. Under these conditions, the complete dataset is represented by:

$$|\varphi\rangle = \frac{1}{\sqrt{N}} \sum_{i=1}^{N-1} \sum_{d=1}^{D-1} x_i^d |d\rangle |i\rangle \tag{20}$$

The vector of (potentially complex-valued) amplitudes is built by concatenating all preprocessed training inputs x_i^d [91]. Again, we can illustrate the encoding procedure with a small example, this time using two data points:

$$\mathbf{x}_1 = \begin{pmatrix} 4 \\ 7 \\ 15 \end{pmatrix}, \quad \mathbf{x}_2 = \begin{pmatrix} 3.5 \\ 1 \\ -5 \end{pmatrix}$$

Note that $\mathbf{x}_2$ also contains float and negative values, which would need more care when using basis encoding. Since we have $N = 2$ data points with $D = 3$ dimensions, we will need $\lceil log_2 (2 * 3) \rceil = \lceil 2.585 \rceil = 3$ qbits. The first task for amplitude encoding is to compute the norm of each sample vector:

$$\|\mathbf{x}_1\| = \sqrt{4^2 + 7^2 + 15^2} = \sqrt{290} \tag{21}$$

$$\|\mathbf{x}_2\| = \sqrt{3.5^2 + 1^2 + (-5)^2} = \sqrt{38.25} \tag{22}$$

Dividing each sample vector by its norm ensures the fulfilment of the unit length condition, which is crucial to obtain a valid encoded quantum state.

$$\mathbf{x}_{1,norm.} = \frac{1}{\sqrt{290}} \begin{pmatrix} 4 \\ 7 \\ 15 \end{pmatrix}, \quad \mathbf{x}_{2,norm.} = \frac{1}{\sqrt{38.25}} \begin{pmatrix} 3.5 \\ 1 \\ -5 \end{pmatrix}$$

To encode the above normalized data points in superposition with amplitude encoding, we need to insert their values in definition (20) for $N = 2$ data points with $D = 3$ dimensions:

$$|\varphi_{\boldsymbol{x}_1,\boldsymbol{x}_2}\rangle = \frac{1}{\sqrt{2}} \left(\sum_{d=0}^{3} x_1^d |d\rangle |0\rangle + \sum_{d=0}^{3} x_2^d |d\rangle |1\rangle \right) \tag{23}$$

where $|d\rangle$ identifies the d-th feature, while $|0\rangle$ and $|1\rangle$ indicate the respective data point. This again confirms that we need $n = 3$ qbits, since we need at least two qbits to distinguish the three features, plus another qbit for the samples. Inserting into (23), the final quantum state representing both $\mathbf{x}_1$ and $\mathbf{x}_2$ in amplitude encoding is given below.

$$\begin{aligned} |\varphi_{\boldsymbol{x}_1,\boldsymbol{x}_2}\rangle = \frac{1}{\sqrt{2}} \bigg(& \frac{4}{\sqrt{290}} |000\rangle + \frac{7}{\sqrt{290}} |010\rangle + \frac{15}{\sqrt{290}} |100\rangle \\ & + \frac{3.5}{\sqrt{38.25}} |001\rangle + \frac{1}{\sqrt{38.25}} |011\rangle - \frac{5}{\sqrt{38.25}} |101\rangle \bigg) \end{aligned} \tag{24}$$

Besides leveraging the basis states and amplitudes in the qbit representation, it turns out there also exists a third class of data encoding that can be considered a compromise between the two former methods: *Angle encoding* [107] in its generic formulation encodes one feature per qbit, but can naturally deal with continuous inputs while requiring a circuit $S(\mathbf{x})$ of only constant depth. This is because the classical inputs are used as angles for a sequence of single-qbit rotation gates (R_X, R_Y, R_Z) acting on the $|0\rangle$-initialized qbit register. As a preprocessing step, all input values should be restricted to a certain interval like $[0,\ \pi]$ [17], $[0,\ \frac{\pi}{2}]$ [106], or $[-\frac{\pi}{2},\ \frac{\pi}{2}]$ [15], since using the full range of $[0,\ 2\pi)$ may lead to a misinterpretation of points at opposite ends of the interval as being similar. In more complex versions of angle encoding which are sometimes called *block encoding*, the Pauli rotations are followed by multi-qbit gates such as *CNOT* to create entanglement between qbits, and this alternation of rotation and entanglement blocks may be repeated several times [17]. A very prominent example for this is the so-called *ZZ Featuremap*, which will be used in the practical examples in Section 6.

The name of the *ZZ Featuremap* already hints at a theoretical result in supervised QML research that also explains why the decision on data encoding is so important: The process of encoding classical data into quantum states is in fact a *feature map* that translates inputs into a higher-dimensional space. Since the inner product as a distance metric is always defined for valid quantum states (see Dirac notation), each such feature map $\phi : \mathcal{X} \rightarrow \mathcal{F}$ can be associated with a *kernel* κ defined by

$$\kappa\left(x, x'\right) = \left\langle \phi\left(x\right), \phi\left(x'\right)\right\rangle_{\mathcal{F}} \tag{25}$$

where $\mathcal{X}$ is the classical input space and $\mathcal{F}$ denotes the higher-dimensional quantum feature space [89]. The benefit of kernels, which have been extensively studied in classical ML, is that we can avoid intractable computations in a potentially infinite-dimensional feature space by instead only computing the kernel function for all combinations of distinct data points $\mathbf{x}$, $\mathbf{x}'$ [44]. With quantum computing resources, this advantage can be extended to the evaluation of kernels that are classically intractable. Moreover, the QML-kernel relation can be formulated in two equally expressive versions: On the one hand, as a *Quantum Support Vector Machine (QSVM)* by replacing the classical kernel with a quantum version, as given by data encoding options; on the other hand, as a *variational model* (VQC for classification, VQR for regression) that learns with only unitary (and therefore linear) operations in the feature space [89]. The difference between the two lies in the training procedure, including factors such as circuit structure and number of circuit evaluations. Finally, the kernel perspective also shows a central difference between *amplitude* and *angle encoding*: Despite the long, complex circuits it requires, the former effectively represents the trivial linear kernel. By contrast, angle encoding introduces a strong non-linearity by mapping data points to products of sums of sine and cosine functions [91]. Since other attempts to characterise ML performance effects of different data encoding classes tend to employ more complex techniques like Fourier coefficient analysis [69], this general observation is certainly one of the most accessible guidelines that have been discovered yet to inform practical QML implementations.

4.3.2 Ansatz

With the data being successfully prepared for quantum computations, the next step in the workflow is to manipulate the quantum state with an **ansatz** $W\left(\theta\right)$, consisting of a sequence of gates, some of which depend on a set of parameters $\{\theta\}$. The term refers to the "template" nature of these quantum circuits. Such a template can be derived from the mathematical structure of the considered problem (*problem-specific ansatz*) or built in a way that keeps errors on NISQ devices on a low level (*hardware-efficient ansatz*) [105]. In the QML literature, ansätze have also been named parameterized quantum circuits [11], variational forms [53], variational circuits [89, 91]. The term of quantum neural networks (QNNs) [19] might refer to the ansatz or the whole variational model. However, the implied analogy to feed-forward neural networks found in classical DL only partly exists. The most obvious similarity is the layer-wise structure that we already got to know when introducing block encoding in the previous step of the workflow. The intuition behind this structure is that alternation of single qbit rotation layers and multi-qbit entangling layers should allow to control the subset of quantum states that can be reached by the model (*expressibility*) as well as the degree to which the model exploits non-local correlations (*entangling capability*) [96]. To provide a visualization how such an ansatz can look like, Figure 10 shows a so-called two-local ansatz, which is an example of a hardware-efficient ansatz.

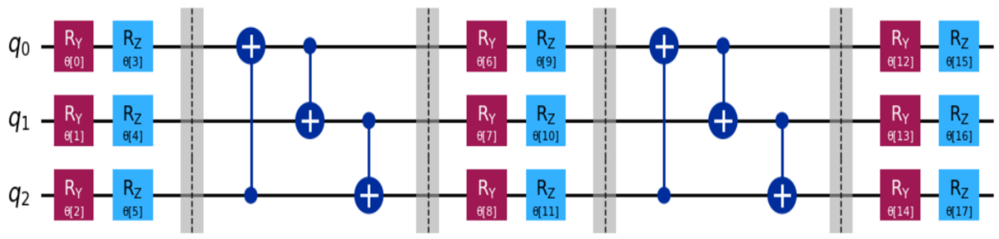

Figure 10: Example for TwoLocal ansatz with 2 repetitions on 3 qbits, created by the author with Qiskit 1.0

The name of this ansatz emphasises that it contains only gates that act on one or two qbits. Structurally, a rotation block is always followed by an entangling block, corresponding to layers that can be repeated. At the end of the circuit, there always is a final rotation block. For the three-qbit example in Figure 10, rotations consist of R_Y and R_Z gates on each qbit, followed by a circular entanglement with $CNOT$ gates. With six rotation gates per layer, there is a total of 18 trainable parameters. As we will see in the next workflow step, the optimization of these parameters can be seen as another similarity between variational QML models and neural networks. However, our example also demonstrates some crucial differences: The ansatz has a constant width, while neural networks usually have a different number of units per layer. But even more importantly, ansätze consist of unitary quantum gates that represent only linear operations, whereas neural networks strongly rely on the non-linearities introduced by activation functions between layers [19]. Thus, it would be more accurate to speak of deep *linear* networks. Apart from data encoding, the only option to introduce non-linearities to such QML models is via special configuration of measurements [91]. Thus, it should be kept in mind that further proposals like quantum convolutional neural networks (QCNN, [23]), quantum generative adversarial networks (QGAN, [25]), or geometric QML models [108] similarly do not just execute their classical counterparts on a quantum computer; they follow the same idea, but try to achieve it with different mechanisms that may or may not be beneficial depending on problem characteristics.

4.3.3 Measurement

With the **measurement** step, we retrieve classical information about the results of running the ansatz on our encoded data. Through the ML lens, measurement provides the prediction or model output, which means that the mathematical formulation of a variational QML model is given by the type of output that we retrieve via measurement [91]. In section 4.1, we have already discussed that the potential measurement outcomes are defined by choosing an observable, and that most multi-qbit observables can be realized by decomposing them into sums of Pauli matrices. However, it is not always the case that all qbits are measured as shown in Figure 9: For variational QML models, there are numerous possibilities for designing the measurement step. For example, a simple setup could use only the first qbit as a *readout qbit*, which is connected to the remaining *data qbits* via controlled operators in the ansatz [17]. When using more than one qbit, the measurement output is often mapped to the desired format with a classical post-processing step. An example for this approach in binary classification settings could be to globally (simultaneously) measure all qbits in the computational basis and then use a *parity function* on the obtained bitstrings to assign an *even* or *odd* class label to data points [101].

For a more systematic association of measurement options with ML theory, we can make the following distinction for the supervised ML paradigm:

- **Deterministic models** are implemented by calculating the *expectation value* of an observable. The result is a prediction of a specific value $f_\theta(\mathbf{x})$. The *readout qbit* approach is an example for this category.
- **Probabilistic models** are equivalent to a *sampling* procedure by performing many measurements. The result is a probability distribution $p_\theta(y|\mathbf{x})$ that may be post-processed, as in the *parity function* example for classification.

For unsupervised ML methods, we can use a similar formulation where measurement results are associated with data points instead of labels [91].

Independent of chosen ML paradigm, the measurement strategy is a central design factor when moving to real quantum hardware, since NISQ devices exhibit varying levels of readout errors [82]. Moreover, global measurements can contribute to an unfavourably shaped optimization landscape [101].

4.3.4 Cost and Optimization

The final part of a single iteration in the variational QML workflow is executed on classical hardware. It consists of two steps that are well-known from DL architectures: Firstly, a **cost function** $C(\theta)$ is evaluated to determine the quality of the result obtained with current parameters. Secondly, an **optimization technique** is used to calculate a new set of parameters for the next iteration, or end the training process in case of convergence to a minimum. While the first step does not pose any new conceptual challenges compared to classical ML, the choice of optimizer merits greater attention. In general, variational QML models can be optimized with either *gradient-based* or *gradient-free* methods [105]. However, the latter, which include Bayesian optimization [29], evolutionary algorithms [28] and genetic algorithms [2], are less commonly applied, mainly in cases where gradient-based methods face some known limitations such as error-induced non-smooth loss functions [11].

For the case of gradient-based optimization, it is instructive to take a look at the chain rule-based formulation for the partial derivative of the cost function C with respect to any of its variational parameters θ_i:

$$\frac{\partial C}{\partial \theta_i} = \frac{\partial C}{\partial f_\theta}\frac{\partial f_\theta}{\partial \theta_i} \tag{26}$$

where f_θ denotes the model output. While the first factor on the right-hand side of the equation is a purely classical computation, the second factor is the partial derivative of a quantum computation, posing the question how this can be evaluated [91]. Here we can distinguish between *numerical* and *analytical* gradients, just as in classical DL theory. The numerical estimation can be done with the *finite difference quotient* given below:

$$\frac{\partial f_\theta}{\partial \theta_i} \approx \frac{f\left(\boldsymbol{\theta} + \epsilon \boldsymbol{e}_i\right) - f\left(\boldsymbol{\theta} - \epsilon \boldsymbol{e}_i\right)}{2\epsilon} \tag{27}$$

where ϵ is a small perturbation factor and $\boldsymbol{e}_i$ is the i-th unit vector [11]. However, the errors when executing the variational circuit with the perturbed parameters on a real device, as well as the approximate nature of the method per se can make its application problematic [91].

Thus, the more practically feasible option is to calculate the analytical gradient of the model output with respect to gate parameters via a *parameter shift rule* [11]. Such a rule can be derived by writing the model output f_θ as expectation value of the chosen observable $\mathcal{M}$. The partial derivative of this expectation value with respect to one of the parameters can then be obtained by executing the variational circuit twice. The most common choice is to shift parameters by $\frac{\pi}{2}$, since this simplifies the derivation via the trigonometric identities $sin\left(\theta \pm \frac{\pi}{2}\right) = \pm cos\left(\theta\right)$ and $cos\left(\theta \pm \frac{\pi}{2}\right) = \mp sin\left(\theta\right)$. In this case, the rule is given by:

$$\frac{\partial \langle \mathcal{M} \rangle_\theta}{\partial \theta_i} = \frac{\langle \mathcal{M} \rangle_{\boldsymbol{\theta}+\frac{\pi}{2}\boldsymbol{e}_i} - \langle \mathcal{M} \rangle_{\boldsymbol{\theta}-\frac{\pi}{2}\boldsymbol{e}_i}}{2} \tag{28}$$

While this expression looks somewhat similar to the finite difference quotient, it is conceptually different, since it estimates the analytical gradient and not the approximate (numerical) gradient [91]. Also note that this procedure, where the whole circuit must be executed twice per parameter, does not scale as well as backpropagation in classical neural networks, where intermediate delta values can be stored [90].

Finally, with the gradients computed, the new set of parameters can be determined with typical DL optimizers, such as plain stochastic gradient descent or Adam. However, for execution on real NISQ hardware, it can be beneficial to use special quantum-aware optimizers [19] which are better equipped to deal with the effects of noise in quantum computations. An example for such an approach is the *quantum natural gradient* [97] that chooses the step size based on local characteristics of the optimization landscape.

4.4 Current Challenges for QML

The research field of variational QML for classical data is still at a nascent stage that does not allow yet to make definitive statements about a supposed superiority over classical ML algorithms in practically relevant applications. To illustrate some promising directions that might bring the community nearer towards such insights, this subsection presents a selection of important theoretical and practical challenges for current QML research.

Firstly, the trainability of variational QML models is often impeded by the occurrence of **barren plateaus** [65] in the optimization landscape. This phenomenon is defined as an exponential concentration of the gradients of the cost function about their mean in the amount of employed qbits. This means that as the number of qbits increases, the gradient distribution forms a narrow bell-shaped curve with a peak at the mean and very long tails representing extremely low gradients. Thus, the optimizer requires an exponential number of measurements to determine the moving direction in the mostly flat cost landscape - otherwise, the optimization steps essentially are a random walk [52]. We can also think of barren plateaus as a consequence of the curse of dimensionality arising from highly expressive models operating in large solution spaces [91]. There are various factors that can individually or collectively lead to barren plateaus and should thus be avoided in model design where possible. These include global measurements, large and deep circuits with many repeated layers, hardware noise, as well as highly entangling data embedding schemes or ansätze [101]. Since seemingly easy strategies like a change of optimization method or the use of error mitigation techniques cannot mitigate the arising problem, it is necessary to focus more on an effective, shallow ansatz circuit structure with a suitable non-random parameter initialization scheme [52].

This insight aligns well with the call for investigations on what the **added value of "quantumness"** in ML models can be from a theoretical perspective. For example, the *inductive bias* of common variational QML models is still poorly understood [15]. This term describes the type of prior knowledge about a problem [19] or restriction of the solution space [10] that is encoded into a learning algorithm and thus influences which data patterns the algorithm will generalize on well. Of course, the dilemma is that we still want to make use of large quantum state spaces in hope of obtaining a beneficial, but classically intractable separation or approximation of the data. Thus, the key to reconcile these conflicting goals might lie in the observation that what matters "[...] is not the size of the space explored by quantum models *per se* but rather *how* the space is used" [52]. Such a perspective is also still lacking for the current practice of integrating variational QML models as quantum layers into hybrid quantum-classical neural network architectures [15]. Nevertheless, there are some results on the utility of quantum specifics that should be seen as an encouragement for future work: an example is the *Quantum No-Free-Lunch theorem* for entangled datasets [93] which demonstrates that moderate entanglement of training samples improves performance of learning algorithms by reducing the size of the required training set. This can be especially beneficial for generalization in cases of high-dimensional data, non-independent features, and instances where data are expensive or scarce.

Talking about performance, this is perhaps the most central challenge for applied variational QML research, which we introduced in Section 4.2

as being driven by **empirical benchmarking**. A natural question is: how reliable are the observations we can make from classically simulated training with only a small number of features? The arising doubts are aggravated by the fact that the *choice* of benchmarks is often driven by trends, leading many QML studies to borrow them from classical ML. However, the down-scaling that is required to use one of the MNIST datasets, for example, can be of such an extent that what remains is essentially a trivial task [15]. In a recent work by Shen et al. [94], this is addressed by providing an approximate encoding of the full Fashion-MNIST dataset [110] which is suitable for simulation and NISQ hardware. While such efforts already constitute an improvement, they cannot replace a more thorough reflection on alternative ways of evaluation that are perhaps more insightful for understanding how to design an efficient model dependent on problem characteristics [90].

A first step in this direction is taken in the benchmarking study by Bowles et al. [15] which uses six artificial binary classification datasets that represent common patterns like linearly separable classes, separating hyperplanes, or polynomial curves in up to 10-dimensional space. These structures both appear in real-world problems and are small enough to serve as simulation inputs on a maximum of 18 qbits. The study uses this setup to compare 12 gradient-based variational QML models with three out-of-the-box classical baselines (MLP, SVM, CNN) by applying a grid-search hyperparameter optimization in 5-fold cross validation. As a result, the classical methods consistently outperform the QML models, and some of the latter even struggle significantly with the relatively simple linear benchmarks. This is in stark contrast to the impression which the reader of QML literature might obtain from the majority of employed benchmarks, which commonly report on superiority of variational QML. Moreover, as Schuld & Killoran [90] note, only few studies explicitly justify their choice of benchmark. Taken together, this points to a lack of reproducibility or **positivity bias** among such publications [15]. Therefore, there is a need for more robust, critically reflected study designs in order to ensure reliability of performance evaluations and to develop new proposals based on discovered limitations.

Finally, we can also take a brief look on some challenges arising from **real hardware usage**. Due to the limited availability, only few studies have been able to observe the implications for variational QML. In general, access to devices is in most cases granted through vendor-specific cloud services, which causes a lack of implementation portability. In addition, it has to be kept in mind that devices typically optimize circuits in a hardware-specific manner, such that results on one device do not carry over to other devices [55]. In fact, the topology of physical qbits can impact performance and the optimal choice of ansatz [16]. Cloud access can also result in very long training times, since each iteration of the variational QML workflow corresponds to a separate job in the hardware queue. To address this issue, Phalak & Ghosh [79] propose a methodology for multi-hardware training to minimize waiting times. However, even this study simulated the effects of its approach and only used real hardware for inference - examples of training on quantum devices, such as for QSVM on a trapped-ion device in Suzuki et al. [99], are very rare. Nevertheless, using quantum hardware for larger problem sizes remains desirable, since scaling might also reveal unexpected challenges such as a lack of classical registers observed in a stress test on a trapped-ion device with a Quantum Neuron Born Machine (QNBM) of increasing complexity [95].

In summary, we have seen that the current state of variational QML methods exhibits four areas of major challenges that indicate directions for future research in this field:

- **Barren plateaus:** Model design factors like deep circuits or global measurements can lead to a mostly flat optimization landscape that impedes trainability.
- **Understanding of inductive biases:** There is a lack of knowledge about the connection between the choice of model design factors and the patterns that the resulting models can learn well.
- **Benchmarking procedures:** Measuring the performance of QML models on benchmarks borrowed from classical ML, downsized by dimensionality reduction, and/or simulated classically only allow limited insights about the true potential of variational QML models.
- **Practical constraints:** Access to real quantum hardware resources is limited and costly. In addition, vendor-specific cloud access can result in long training durations due to waiting times and may lead to implementation portability issues.

5 Review of QML Business Applications

This section examines the extent to which QML methods have been applied to or specifically developed for the business use cases established in Section 3 and related scenarios. This use case-centric approach has previously been leveraged by two related works focusing on the finance domain: Mironowicz et al. [66] review QML methods for portfolio optimization, market prediction and trading, pricing, and risk management. Another survey by Pistoia et al. [80] primarily distinguishes between regression, classification, clustering, generative, feature extraction, Reinforcement Learning, and NLP techniques. On the second hierarchical level, each of these ML paradigms is associated with several financial use cases. However, a more comprehensive review of QML applications in a broader range of organizational activities is still missing in the literature. Moreover, the two above-mentioned publications include many use cases of quantum annealers, while the present review is exclusively dedicated to QML methods designed for gate-based quantum computers.

5.1 Review Results

The literature review (see Section 2.2 for methodology) yielded 26 relevant references for 13 business use cases, among which 11 categories also appeared in the review of Data Science Labs in Section 3. Two of the use cases among the Lab projects (*Data Collection and Enrichment* and *Medical*) were excluded for the present review, since they do not belong to typical business domains. For the remaining 8 out of 21 Lab use cases, no relevant publications could be identified in the review process. However, this does not necessarily mean that their characteristics and requirements prevent the usage of QML methods; it rather points at the fact that no QML experiments on real-world or proxy datasets belonging to these problem classes have been performed and reported in the literature.

Table 2 provides an overview on the results of the review process, listing the ML paradigms, references, and applied QML algorithms for each use case. Regarding the **distribution of business domains**, we can overall confirm the relatively high prominence of finance applications, with 14 out of 26 references belonging to financial use cases (*Credit Risk Management*, *Demand and Price Prediction*, *Fraud Detection*, *Portfolio Optimization*, and *Valuation*). Conversely, this means that almost half of the publications tackle other corporate functions like Marketing, Logistics, or Information Systems, indicating that QML is also gaining attention outside the financial context.

Table 2: Overview of identified literature on application of (gate-based) QML methods to business use cases

Business Use Case	**ML Paradigms**	**References**	**QML Algorithms**
Applied Environmental Research	Supervised Learning, Time Series	Cao et al., 2023 [18] Kumar et al., 2023 [51] Sagingalieva et al., 2023 [86]	Hybrid linear-layer enhanced Q-LSTM CNOT-gate induced NN Hybrid Q-LSTM, Hybrid Seq2Seq NN
Churn Prediction	Supervised Learning, Time Series	Thakkar et al., 2024 [100]	Random Forest with improved sampling via quantum Determinantal Point Processes (DPP)
Credit Risk Management	Supervised Learning, Time Series	Mancilla & Pere, 2022 [61] Mancilla et al., 2024 [62] Schetakis et al., 2024 [88] Thakkar et al., 2024 [100]	QSVM, VQC Systemic Quantum Score (SQS) for QSVM Hybrid QNN QNN with orthogonal and compound layers
Demand and Price Prediction	Unsupervised Learning, Supervised Learning, Time Series	Orlandi et al., 2024 [73] Paquet&Soleymani, 2022 [76] Stühler et al., 2024 [98]	Q-Wasserstein GAN with Gradient Penalty Hybrid Deep QNN QSVR with fidelity and projected q-kernels
Fraud Detection	Supervised Learning, Time Series	Grossi et al., 2022 [35] Innan et al., 2023 [41] Innan et al., 2024 [42] Yalovetzky et al., 2024 [111]	QSVM, hybrid ensemble model QSVC, VCQ, SamplerQNN, EstimatorQNN Quantum Graph NN QC-Forest-Retrain for multi-class RF
IoT Application	Supervised Learning, Time Series	Herbst et al., 2024 [37]	VQR for Streaming Data
Logistics Automation and Analytics	Reinforcement Learning, Supervised Learning	Correll et al., 2023 [24] Jahin et al., 2023 [43]	Reinforcement Learning with quantum orthogonal NN and hybrid attention Hybrid QNN

Business Use Case	**ML Paradigms**	**References**	**QML Algorithms**
(Other) Marketing Analytics	Supervised Learning, Time Series	Gandhudi et al., 2023 [34]	Parameterized quantum stochastic gradient descent model
Portfolio Optimiza-tion	Unsupervised Learning, Generative Modelling	Alcazar et al., 2020 [4] Lee et al., 2023 [54]	Quantum Circuit Born machine (QCBM) Quantum GAN (Info-QGAN)
Predictive Mainte-nance	Supervised Learning, Time Series	Maior et al., 2023 [59]	Parameterized VQE circuits connected to classical MLP
Process Analytics	Supervised Learning	Hill et al., 2023 [39]	QSVM
Recom-mender System	Unsupervised Learning, Recom-mender Algorithms	Li et al., 2024 [56] Ouedrhiri et al., 2022 [74]	Quantum Nearest Neighbor Collaborative Filtering Algorithm Adapted quantum k-means clustering
Valuation	Unsupervised Learning, Generative Modelling	Riofrio et al., 2024 [85]	QCBM, QGAN

In terms of **ML paradigms**, most use cases are implemented as Supervised ML tasks, and almost all of these cases work with time series data, rather than cross-sectional data. Given the probabilistic nature of quantum computing, it can be argued that it is a sensible choice to focus on time series, where gaining an understanding of complex random processes is a frequent goal of analysis. However, there are several notable business applications of Unsupervised ML as well. In particular, generative modelling is regarded by parts of the QML community as most promising candidate for a near-term practical quantum advantage, as opposed to a more rigorous provable advantage which is more difficult to establish [38]. These methods, including Quantum Circuit Born Machine (QCBM, [57]) and Quantum

Generative Adversarial Networks (QGAN, [25]) as prominent examples, have been utilized for *Portfolio Optimization* by generating portfolio return distributions [4, 54], as well as for proxy scenarios that could be applicable in a valuation process [85]. In addition, it has been demonstrated that *Demand and Price Prediction* can be improved by learning a synthetic representation of financial time series with a QGAN [73]. Furthermore, the Unsupervised paradigm includes two proposals for quantum *Recommender System* algorithms with speedups compared to existing classical methods confirmed by experimental validation on publicly available datasets: while Ouedrhiri et al. [74] show a logarithmic reduction in time complexity in an adaptation of quantum k-means clustering, the paper by Li et al. [56] introduces a quantum nearest-neighbor algorithm with exponential speedup of some steps by using quantum phase estimation and quantum Fourier transform subroutines. However, it has to be mentioned that the full impact of these speedups is dependent on the realization of better error mitigation on quantum hardware. Lastly, Correll et al. [24] cover an ML paradigm that did not appear in the Data Science Lab review in Section 3: By combining supervised quantum orthogonal neural networks and a hybrid quantum attention mechanism with a Reinforcement Learning architecture, they tackle a real-world vehicle routing problem and achieve results that are comparable to human track assignment. Their approach demonstrates that QML methods fulfilling a specific task can seamlessly fit in a larger ML workflow combining multiple approaches.

A further notable aspect of the study by Correll et al. [24] is the **usage of real quantum hardware**. The paper reports on experiments conducted on an 11-qbit QCWare IonQ quantum computer, highlighting the benefits of relatively low noise level and physical all-to-all qbit connectivity on trapped-ion quantum devices. The results show high quality solutions to the truck routing problem for implementation on up to 6 qbits, but also deteriorating quality for higher numbers of qbits due to the noise. The only other reference that performed computations on a gate-based quantum device is the paper by Thakkar et al. [100], which reports on two distinct QML approaches for the *Churn Prediction* and *Credit Risk Management* use cases with real-world data from a Latin-American bank. In contrast to the previous paper, this reference uses IBM superconducting quantum hardware, which offers more qbits and speed at the cost of more frequent errors and only nearest-neighbor physical connectivity. Firstly, the study proposes a methodology to improve the performance of Random Forest on the churn prediction task by replacing the uniform sampling to build the individual trees in the original algorithm with an alternative procedure: Determinantal Point Processes (DPP), which are "probabilistic models that can be used to sample diverse subsets of items from a larger set" [100], can be implemented on quantum computers with sub-polynomial time complexity in the number of features under certain conditions. In a classical simulation, the more diverse DPP samples lead to a 6% increase in precision for the Random Forest model. For a reduced problem size, DPP is also performed on a 16-qbit quantum device, finding deterioration of performance with increasing problem dimension, similar to Correll et al.'s results. For the second task of credit risk prediction, Thakkar et al. implement QNN architectures with several types of layers by combining training on simulators with inference on a 27-qbit quantum device. For the orthogonal layer QNN, the application of advanced circuit optimization and error mitigation techniques allows for an almost noiseless inference, highlighting the feasibility of this architecture on NISQ hardware. However, it is important to note that scaling to larger circuit size might still be difficult due to the overhead introduced by more complex transpilation and error mitigation [100].

Proceeding with **financial use cases**, *Credit Risk Management* has received considerable attention from a variety of QML perspectives. Mancilla & Pere [61] address the fact that high-dimensional datasets need to be downscaled for QML algorithms (QSVM, VQC) by investigating the impact of dimensionality reduction techniques on the performance of out-of-the-box PennyLane and Qiskit classifiers, finding Linear Discriminant Analysis (LDA) to yield the best results on the UCI Credit Card dataset [113]. Next, Mancilla et al. [62] tackle the efficient design of quantum kernels for classification by an evolutionary, gradient-free algorithm. When applied to the credit risk data of a Spanish fintech start-up, the resulting QSVM classifier shows superior AUC generalization performance over classical SVM and XGBoost in a regime of data-scarcity. Another application on real-world data from Singaporian SMEs is presented in Schetakis et al. [88]. Their hybrid QNN consists of classical feed-forward layers for the inputs, a PennyLane quantum layer with angle embedding, entangling layers, and measurement, as well as a final classical decision layer for the default prediction outcome.

Turning to *Fraud Detection* as the second larger category of financial use cases, we again find a proposal to improve the classical Random Forest algorithm, this time regarding the need for retraining in the context of large amounts of streaming data: Yalovetzky et al. [111] introduce the QC-Forest algorithm for building and retraining the Random Forest in a multi-class classification setting, as commonly found with transaction data. However, only a classical proxy version of the algorithm is tested, demonstrating lower retraining time while maintaining performance as in the original Random

Forest version. A more readily implementable approach for fraud prediction problems is taken by Grossi et al. [35] who demonstrate how to combine out-of-the-box classical and quantum models into a hybrid ensemble model. In their implementation, a meta-classifier is trained to produce the final decision if the individual models disagree. The reported simulation results show that the inclusion of a relatively simple quantum model like QSVM with access to few features can improve performance in combination with a classical model like XGBoost with access to all features. This highlights that classical and quantum methods may complement each other by detecting different patterns. A similarly practice-driven investigation is delivered in Innan et al. (2023, [41]) by applying several combinations of feature maps and classification algorithms provided by the Qiskit Machine Learning library on a balanced synthetic dataset from the BankSim simulator [58]. Innan et al. (2024, [42]) take a more complex, novel approach with a multi-layer Quantum Graph Neural Network (QGNN). To use transaction data as graph inputs, topological data analysis and clustering are applied to generate the nodes and edges that together represent a given labelled transaction. The QGNN trained with Qiskit on 6 qbits achieved better AUC and accuracy compared to the classical GNN on the real-world ULB fraud detection benchmark.

Financial *Demand and Price Prediction* tasks as a typical example for the Time Series paradigm have recently also been formulated in the Supervised QML framework. Paquet & Soleymani [76] convert the time series of securities prices to a sequence of density matrices, which is the more general way of representing quantum states compared to the state vectors used as inputs to most QML algorithms. A hybrid deep QNN is used to predict the density matrices at a given later point in time, and predicted price is then obtained via a learnable measurement operation. Application of the method yielded relative errors below 1% for 23 out of 24 NASDAQ stocks. However, the authors report a total training time of approximately 24 hours for all stocks together, showing the high resource requirements when simulating these quantum computations. By contrast, Stühler et al. [98] use an Autoencoder to obtain a simplified representation of their dataset that was obtained from online construction equipment market portals. The encoded features are fed into a Support Vector Regression with several different quantum kernels: while fidelity kernels operate in the full quantum Hilbert space, projected kernels map back into classical space to reduce expressivity and thus avoid the curse of dimensionality. Since these quantum kernels show competitive performance, they can indeed be seen as a valuable addition to the typically used linear, Gaussian, or Radial Basis Function (RBF) kernels.

The three references identified for the *Applied Environmental Research* category are unified by their characteristics as **sustainability-directed use cases**, but from a methodological perspective, they also share the forecasting approach with the *Demand and Price Prediction* cases. Cao et al. [18] implement a hybrid Q-LSTM with classical linear layers before and after the variational quantum circuit to predict carbon prices retrieved from the European Union Emission Trading system. While the model shows a clear improvement over the pure Q-LSTM, the prediction performance very closely follows the results of the classical LSTM, making it questionable whether the quantum part plays a vital role in the network. By contrast, the smart grid load forecasting algorithm introduced in Kumar et al. [51] shows considerable improvements over competing state-of-the-art models of up to 78.92% and 83.61% in terms of RMSE and MAE, respectively. The proposed QNN uses the CNOT gate as an activation function between layers and is trained in a gradient-free, evolutionary manner. Another QML forecasting application is delivered by Sagingalieva et al. [86] who propose three hybrid models to predict the power production of photovoltaic panels: A hybrid QNN with standard variational repetitive quantum layer and ReLU activation, a hybrid Q-LSTM with a Quantum Depth-Infused (QDI) layer on each LSTM gate, and a hybrid quantum sequence-to-sequence (Seq2Seq) model which consists of two LSTMs with a final quantum layer. While the first two take a 24 hour timeframe as input to make a prediction for the subsequent hour, the Seq2Seq model allows for arbitrary length of input and output time series. For each model, the authors report a lower MSE compared to its classical counterpart. This observation still holds when gradually reducing dataset size, again indicating the utility of QML in settings of data scarcity. The paper by Herbst et al. [37] which implements an *IoT Application* use case can also be argued to belong to the field of sustainability, since the UCI bike sharing dataset [32] used to illustrate their approach contains environmental data typically registered by IoT sensors. The study delivers a first intuition how a future implementation of Quantum Edge Analytics could look like, where data streamed from IoT devices are encoded and processed on an error-corrected quantum device directly on edge nodes before delivering results to users.

For the area of **Logistics and Operations Management**, gate-based QML publications belonging to three use cases have been reviewed. The Reinforcement Learning approach of Correll et al. [24] in *Logistics Automation and Analytics* has already been mentioned. In the same category, Jahin et al. [43] developed QAmplifyNet, a hybrid QNN for supply chain backorder prediction. The model design is inspired by the principle of amplitude amplification, which is used in Grover's algorithm [36] for unstructured quantum search, for example: this algorithmic technique increases the probability of measuring a desired basis state (e.g., the searched element) by iteratively increasing its amplitude while reducing the amplitudes of undesired states. The study includes a comprehensive benchmarking procedure where the proposed architecture was trained on two GPUs and achieved "superiority over eight classical models, three classically stacked quantum ensembles, five quantum neural networks, and a deep reinforcement learning model" [43]. Furthermore, the authors utilize local and global interpretability techniques [68] to investigate the inner workings of the model. Since the quantum layer in QAmplifyNet works with only 2 qbits, this model can be regarded as highly feasible for real-world application to mitigate the potentially high cost impact of stock depletion in supply chain inventory management.

The *Predictive Maintenance* use case implemented by Maior et al. [59] also leverages a characteristic aspect of quantum computing, namely the relation to trigonometric functions, as demonstrated by rotations on the Bloch Sphere of a single qbit, for example. By choosing to work with vibration signals of rotating machinery from two machine prognostics and health management databases as inputs, the paper delivers a case study where potential QML utility can already be argued based on the properties of the data. In their architecture, Maior et al. use angle encoding to feed different combinations of features like mean, variance, kurtosis, or maximum amplitude of the signals into a parameterized quantum circuit consisting of Variational Quantum Eigensolver (VQE) and various gate combinations. The classical results of measuring circuit outputs are subsequently used as inputs to a standard Multi-Layer Perceptron (MLP) performing multi-class classification of machine health states. The experiments yield consistently better results for the hybrid approach than for the MLP alone on both datasets.

The *Process Analytics* case presented by Hill et al. [39] shares the goal of predictive monitoring, but considers event log data, as typically encountered in Business Process Management Systems (BPMS). Moreover, the reference focuses on inter-case features like number of peer cases, number of working resources, most frequent activity, or average delay within a given time window to predict labels like next activity or remaining time for a set of cases. However, due to computational constraints of simulating QSVM, the experiments only contain a maximum of two inter-case features. Since QSVM outperforms XGBoost in accuracy for 5 out of 10 runs, the authors conclude that this QML algorithm is competitive for their business use case.

Finally, **Marketing** applications have received the comparatively least attention in the QML literature. Apart from the *Churn Prediction* approach implemented on quantum hardware in Thakkar et al. [100] that was described earlier in this section, the only other reference that performs *(Other) Marketing Analytics* with QML methods is Gandhudi et al. [34]. In this work, a Parameterized Quantum Stochastic Gradient Descent (PQSDG) model is developed and tested in a causal modelling framework for media mix data. Features of the employed dataset include metrics like spending, clicks, and queries for different media channels, as well brand and competitor sales. These are subjected to various transformations (log transform, exponential rolling average, normalization) and arranged in a causal directed acyclic graph to enable statements about causality. For both conducted experiments, the proposed PQSDG achieves better MSE values compared to a set of other QML approaches including Quantum Decision Tree, Quantum Random Forest, and Quantum LightGBM.

5.2 Discussion

The results of the above review clearly align with the challenges for current QML research presented in Section 4.4. Firstly, a clear argumentation why application of QML methods could be advantageous based on problem characteristics is missing for a significant share of the mentioned studies. This is particularly evident for cases where variational models are incorporated into classical DL architectures as a quantum layer. A more explicit justification can be seen in setups which aim to exploit more general characteristics of quantum computers: Examples include their capability of generating diverse samples [100] and the principle of amplitude amplification [43]. Another major issue is the choice of benchmarking procedure that led to the positive results that are presented in most references. For studies that only report results on one or two evaluation metrics, doubts arise whether these are methodologically robust findings. Moreover, a broader evaluation scheme, paired with transparency on employed model design principles and reasoning in choosing classical baselines, can also help practitioners to better understand trade-offs implied by utilization of QML methods. Finally, the most critical factor that limits statements about real-world business utility of QML is the need for down-scaling problem sizes, which applies both for the usual choice of execution on simulators and the relatively rare experiments on current NISQ devices. In this respect, only time can tell when the state of quantum computing will allow for problems at realistic scale, but QML research can do its share by continuing to strive for more methodologically sound and practically insightful study designs.

6 Results of QML Example Implementations

To complement the review of existing works on QML business applications in the previous section, we present two example implementations on publicly available proxy datasets representing use cases that occurred in the review. While the overview provided here is restricted to the main results, the accompanying notebooks are much more detailed in explaining the model design factors in the coding workflow (see **Appendix B**).

6.1 Churn Prediction with PennyLane

This first case is implemented in the special-purpose QML framework PennyLane [12], partially adapting code from Combarro & Gonzalez-Castillo [22]. The dataset [102] emulates a typical scenario of predicting customer churn in retail banking. It contains 10.000 observations with 9 customer features and a binary target indicating churn. Some descriptive statistics are provided in Table 3 below.

Table 3: Descriptive statistics for churn prediction dataset

Feature	Mean	Std	Min	Max
credit score	650.53	96.65	350.00	850.00
age	38.92	10.49	18.00	92.00
tenure	5.01	2.89	0.00	10.00
balance	76485.89	62397.41	0.00	250898.09
no. of products	1.53	0.58	1.00	4.00
credit card	0.71	0.46	0.00	1.00
active member	0.52	0.50	0.00	1.00
estimated salary	100090.24	57510.49	11.58	199992.48
churn	0.20	0.40	0.00	1.00
male	0.55	0.50	0.00	1.00

Since the features in the dataset differ greatly in their means and standard deviations, we will need to standardize them in order to achieve better results with our predictions. Checking the balance between classes of the target reveals that 7963 samples are non-churners and 2037 are churners. Since the business goal is to predict the churning correctly, this imbalance could be addressed with methods such as SMOTE [20]. However, the focus here is to demonstrate the QML workflow instead of aiming for optimal performance.

The **pre-processing** step consists of an 80-10-10% train-validation-test split and standardization via feature scaling: Min-max scaling ensures that all features are restricted to the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$. Next, we **build a variational QML model (QNN)** with PennyLane following the familiar workflow from Section 4.3. For the data encoding step, we use the *ZZFeatureMap*, which encodes classical features in single qbit rotations and pairwise ZZ interactions. For the case of 4 qbits, this means we can parameterize 4 operations on individual qbits and 6 operations on qbit pairs, making a total of 10 operations available for encoding. Thus, we will use 4 qbits to represent our 9 customer features. The model itself consists of a *TwoLocal ansatz* with R_Y rotations, $CNOT$ entanglement and two repetitions. To retrieve model outputs, we need to choose an observable with eigenvalues matching the class labels $\{0, 1\}$:

$$M = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

The measurement strategy consists of estimating the expectation value of this observable on the first qbit in the model's circuit (readout qbit). Cost is evaluated with binary cross-entropy loss and optimized via Adam optimizer with a moderate learning rate of 0.001 and early stopping.

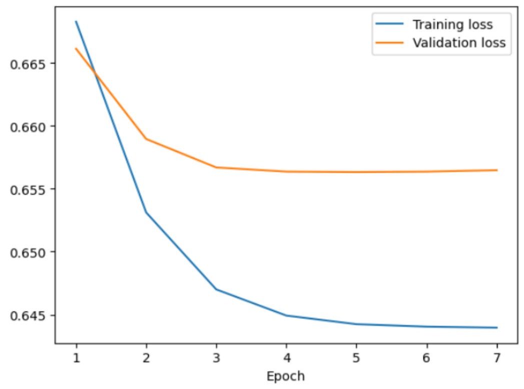

Figure 11: Train and validation loss history of PennyLane QNN on the churn prediction task

From an implementation perspective, the above design factors are bound together by defining a PennyLane QuantumNode on the chosen simulator (PennyLane Lightning). An advantage with PennyLane is that model building will seem familiar to experienced users of Deep Learning frameworks. In this example, the TensorFlow/Keras interface is used: Essentially, we build a feed-forward neural network with the Keras Sequential pattern, using the variational quantum circuit as the only layer. The batch size is set to 50 to allow for some parallel processing and stochasticity in gradient descent while keeping computational load at a manageable level. The training process stopped early after 7 out of 10 epochs and is visualized in terms of loss history in Figure 11.

Table 4: Evaluation metrics for sklearn Random Forest and PennyLane QNN on churn prediction test set

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.849	0.757	0.414	0.535
PennyLane QNN	0.659	0.243	0.295	0.267

Finally, the QNN is evaluated on the test set against a sklearn Random Forest baseline with default settings. As seen from Table 4, the classical model clearly performs better on all metrics. Considering the business logics of churn prediction, it is especially worthwhile to consider the precision and recall metrics, since these indicate how well the models can deal with the positive minority class we are interested in. The results show that the out-of-the-box Random Forest also shows some deficits here, since it misses 60% of the churners (recall). However, this is still a lot better than the QNN, which only correctly predicts about 30% of churners and on top of this makes a lot of false positive predictions (precision of about 24%). Of course, a real-world implementation would typically also try specific methods for dealing with class imbalances and can then be expected to yield better results. In addition, the two models did not go through a hyperparameter optimization procedure. Nevertheless, the comparison is taking place on common grounds, demonstrating that using variational QML models in an out-of-the-box manner does typically leads to worse results as opposed to established classical ML algorithms.

6.2 Machine Failure Classification with Qiskit Machine Learning

The second example follows a very similar workflow, but this time implemented using the Qiskit Machine Learning library belonging to the Qiskit 1.0 SDK [45]. Furthermore, the implementation includes two QML classifiers from this framework that allow for different degree of customization.

The chosen dataset consists of synthetic data for machine failure prediction [64]. Since Qiskit at the time of writing does not natively support processing batches of samples, the size of the dataset is reduced from originally 10.000 samples to 1.000 samples. The undersampling procedure is designed to keep all the rare failure cases to reduce class imbalance, resulting in a total of 661 non-failure and 339 failure cases.

Table 5: Descriptive statistics for the predictive maintenance dataset reduced by undersampling

Feature	Mean	Std	Min	Max
air temperature [K]	300.26	2.00	295.60	304.40
process temperature [K]	310.11	1.42	306.10	313.70
rotational speed [rpm]	1521.56	259.55	1181.00	2886.00
torque [Nm]	43.36	13.06	3.80	76.60
tool wear [min]	118.02	69.32	0.00	253.00
target	0.34	0.47	0.00	1.00

In addition to the binary target and 5 numerical features summarized in Table 5, there is one more categorical feature *Type* that takes 3 levels *H*, *L*, and *M*. As a first **pre-processing** step, the original categorical feature is replaced by three features *Type H*, *Type L*, *Type M* using $\{-1, 1\}$. Since the pre-built convenience implementation for binary classification in the Qiskit ML library that will serve as first model uses Pauli Z for measurement, we also need to recode the target values to be $\{-1, 1\}$. Similar to the churn prediction example, we again perform an 80-10-10% train-validation-test split and min-max-scaling to the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

A benefit of Qiskit for **model building** is the great variety of circuit templates which can be used instead of defining each gate by hand. In both models for this example, we use the *ZZFeatureMap* template on 8 qbits for data encoding to experiment with a more complex quantum state representation compared to the Churn Prediction example with 4 qbits in the previous section: the *ZZFeatureMap* encodes each of the 8 classical features on an individual qbit and uses the 28 unique qbit pairs to represent feature interactions. Regarding the ansatz, we apply the *TwoLocal* template with a setting of two repetitions of R_Y and RZ rotations and circular $CNOT$

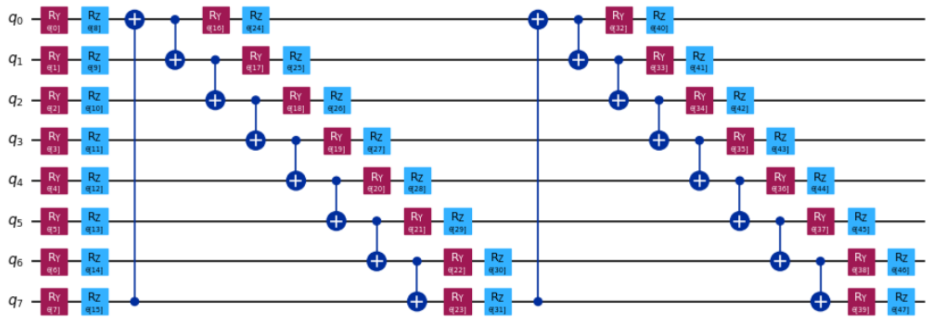
q0 RY θ[0] RZ θ[8] RY θ[16] RZ θ[24] RY θ[32] RZ θ[40]
q1 RY θ[1] RZ θ[9] RY θ[17] RZ θ[25] RY θ[33] RZ θ[41]
q2 RY θ[2] RZ θ[10] RY θ[18] RZ θ[26] RY θ[34] RZ θ[42]
q3 RY θ[3] RZ θ[11] RY θ[19] RZ θ[27] RY θ[35] RZ θ[43]
q4 RY θ[4] RZ θ[12] RY θ[20] RZ θ[28] RY θ[36] RZ θ[44]
q5 RY θ[5] RZ θ[13] RY θ[21] RZ θ[29] RY θ[37] RZ θ[45]
q6 RY θ[6] RZ θ[14] RY θ[22] RZ θ[30] RY θ[38] RZ θ[46]
q7 RY θ[7] RZ θ[15] RY θ[23] RZ θ[31] RY θ[39] RZ θ[47]

Figure 12: Qiskit TwoLocal ansatz for VQC and NeuralNetworkClassifier

entanglement, as depicted in Figure 12. In model 1, the convenience implementation *VariationalQuantumClassifier* (VQC), all that it takes to run the resulting variational model on the default simulator (Sampler primitive) is a choice of cost function (binary cross-entropy loss) and optimizer (COBYLA with a maximum of 30 iterations), followed by calling the *fit()* method on the training set.

By contrast, model 2, the *NeuralNetworkClassifier* allows for customizing aspects such as the observable, measurement via sampling or expectation values, initialization of variational parameters, and structure of the training loop. Thus, we can use this model to showcase the effects of parameter initialization techniques, which represent one way how to address the trainability problems arising from the barren plateau phenomenon. Inspired by the analogous practices in DL models, the study by Kashif et al. [47] compared the effects of LeCun, Xavier, He, and Orthogonal initialization techniques to random initialization, finding a 62.3% reduction in gradient variance decay for Xavier initialization as best-performing method. Thus, we use Xavier normal initialization for our NeuralNetworkClassifier. In this method, the weights are drawn from following normal distribution:

$$\theta \sim \mathcal{N}\left(0, \frac{2}{n_{in} + n_{out}}\right) \tag{29}$$

where n_{in}, n_{out} stand for the number of incoming and outgoing connections, respectively [47]. In our case, both are equal to the number of qbits, since we have a circuit of constant width. By defining a custom training function, the samples drawn from the distribution in (22) can be passed as initial points to the optimizer. The only other changes in model 2 are the usage of Pauli Z expectation values from the first qbit (readout qbit) as measurement strategy and the choice of the Qiskit Aer simulator; all other settings regarding circuit, cost, and optimization remain the same as in model 1 with the *VariationalQuantumClassifier* as high-level method.

Table 6: Evaluation metrics for Random Forest, Qiskit VQC and NeuralNetworkClassifier on predictive maintenance test set

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Random Forest	0.910	0.821	0.941	0.877
VQC	0.510	0.358	0.559	0.437
NeuralNetwork-Classifier	0.580	0.436	0.706	0.539

The **evaluation** of both models again takes a sklearn Random Forest with default settings as a classical baseline. From the results in Table 6, we can see that the classical model is superior in every way. Its only limitation from a business perspective is that it still performs relatively worse on the positive class, which we would like to identify correctly. This pattern can also be observed for both QML models. In relation to this, we would be especially interested in high recall, keeping the business logic of the case in mind. Focusing on this metric, we can see that Random Forest identified almost all positive samples (94.1%), followed by the NeuralNetworkClassifier (70.6%) and VQC (55.9%). Thus, we can see that using a more customized QML model with a deliberately chosen paramater initialization strategy paid off with an almost 15% increase in recall compared to the default implementation.

As a limitation, it has to be mentioned that the two models presented here are only a prototypical demonstration on how to conduct variational QML with Qiskit. Potential **next steps** could be to try other data encodings or ansätze, to compare the effects of using other optimizers or measurement strategies, or to implement the other parameter initialization options from Kashif et al. [47]. Furthermore, Qiskit ML offers a connector to PyTorch which enables larger problem sizes and building hybrid quantum-classical models.

6.3 Personal Insights on PennyLane versus Qiskit ML

While the tools utilized for the above business application proxy cases share many similar features, their practical usage also revealed some differences. The latter emerge from the fact that both programming frameworks were built with different goals in mind: On the one hand, PennyLane primarily aims to deliver optimal QML performance with a design-approach taken from well-known DL frameworks, even if it also includes all the necessary functionalities for building general quantum algorithms. On the other hand, Qiskit ML is a relatively small part of the Qiskit framework, which has its focus on implementing transpilation and error mitigation techniques that enable optimal use of current quantum hardware. For example, tools like the Qiskit Aer simulator and so-called fake backends allow advanced noise modelling that gives an impression of the quality of results that can be expected from a certain real quantum device. PennyLane itself does not offer such functionalities, but can be connected to Qiskit or other frameworks for such purposes. Moreover, the simulators offered by PennyLane also allow to move computations to GPUs, including automatic selection of the best simulation option to enable larger problem sizes.

Regarding the **model building** process, Qiskit ML implementations like VQC or NeuralNetworkClassifier exhibit a structure that resembles the approach of the Python sklearn library. Integration into hybrid architectures is only possible via PyTorch. By contrast, PennyLane follows a more flexible structure where several layers of the variational model are bound together with the QNode abstraction, such that even the structure of the model could be trained. The design approach is comparable to DL frameworks, which is underscored by its TensorFlow, PyTorch, and JAX interfaces for automatic differentiation. Overall, it can be argued that Qiskit ML offers a range of more intuitive, conveniently usable methods that are ideal for practically learning about the variational QML workflow. For more experienced users, DL enthusiasts and corporate applications, the flexibility of PennyLane can be seen as the more attractive option.

7 Conclusion

Corporate Data Science teams typically face two major challenges: Firstly, the task of problem formulation, where business goals and real-world entities are transformed to a numerical representation; secondly, the increasing resource consumption of ML training processes directed at improving generalization. Motivated by the capabilities and starting commercial availability of quantum computers and combining theoretical and practical aspects, the present thesis analysed how the application of Quantum Machine Learning (QML) can potentially address these challenges.

With a review of student projects conducted in cooperation with corporate partners in the Data Science Lab courses at Vienna University of Economics and Business, we established a mapping of business use cases, classical ML implementations, and evaluation metrics that serves as a representation of common problem formulations across corporate domains. Next, we reviewed essential theoretical principles of quantum computing and showed how these building blocks can be leveraged to build a hybrid quantum-classical workflow for learning tasks. Taking a closer look at these variational QML methods, we also identified their current limitations, which include trainability issues, a lack of theoretical understanding of the added value of "quantumness", access to real quantum hardware, and the design of valid benchmarks.

With a review of gate-based QML applications to a broad range business use cases that was previously missing in the literature, we found these challenges confirmed, but also discovered that QML architectures with well thought-out design decisions can be a viable and beneficial alternative to classical ML techniques. Finally, practical implementation work using PennyLane and the Qiskit ML library demonstrated how to run variational QML algorithms on two publicly available proxy datasets for churn prediction and predictive maintenance. The experiments revealed classical models as clearly superior over their quantum counterparts when applied in an out-of-the-box manner. Furthermore, we gained qualitative insights on the relative advantages of both QML frameworks.

Overall, the QML research field, even if still at a nascent stage, shows some promising results towards reducing ML training effort and generating more generalizable predictions. However, future research on QML business applications could gain particularly beneficial insights by focusing more on the first challenge of problem formulation: By comparing typical statistical properties occurring in different use cases and linking them with QML model characteristics, we could better understand where QML algorithms will more likely deliver business value.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, ..., and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. *arXiv preprint arXiv:1603.04467v2*, 2015. Software available from `https://www.tensorflow.org/`.

[2] Giovanni Acampora, Angela Chiatto, and Autilia Vitiello. Training variational quantum circuits through genetic algorithms. In *2022 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2022.

[3] Tameem Albash and Daniel A. Lidar. Adiabatic quantum computation. *Reviews of Modern Physics*, 90(1):015002, 2018.

[4] Javier Alcazar, Vicente Leyton-Ortega, and Alejandro Perdomo-Ortiz. Classical versus quantum models in machine learning: insights from a finance application. *Machine Learning: Science and Technology*, 1(3):035003, 2020.

[5] Maha Alkhayrat, Mohamad Aljnidi, and Kadan Aljoumaa. A comparative dimensionality reduction study in telecom customer segmentation using deep learning and pca. *Journal of Big Data*, 7:1–23, 2020.

[6] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, ..., and Soumith Chintala. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS '24, page 929–947. Association for Computing Machinery, 2024.

[7] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, ..., and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[8] Sheldon Axler. *Linear Algebra Done Right.* Springer Nature, Cham, 2024.

[9] Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *arXiv preprint arXiv:2102.06701v2*, 2024.

[10] Leonardo Banchi, Jason Pereira, and Stefano Pirandola. Generalization in quantum machine learning: A quantum information standpoint. *PRX Quantum*, 2(4):040321, 2021.

[11] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, 2019.

[12] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Shahnawaz Ahmed, Vishnu Ajith, ..., and Nathan Killoran. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.

[13] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[14] Steven Bird, Edward Loper, and Ewan Klein. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit.* O'Reilly Media Inc., 2009.

[15] Joseph Bowles, Shahnawaz Ahmed, and Maria Schuld. Better than classical? the subtle art of benchmarking quantum machine learning models. *arXiv preprint arXiv:2403.07059*, 2024.

[16] Giuseppe Buonaiuto, Francesco Gargiulo, Giuseppe De Pietro, Massimo Esposito, and Marco Pota. The effects of quantum hardware properties on the performances of variational quantum learning algorithms. *Quantum Machine Intelligence*, 6(1):9, 2024.

[17] Dominic Byrne, Matthew G. Cook, and Ethan N. Evans. A quick introduction to quantum machine learning for non-practitioners. *arXiv preprint arXiv:2402.14694*, 2024.

[18] Yuji Cao, Xiyuan Zhou, Xiang Fei, Huan Zhao, Wenxuan Liu, and Junhua Zhao. Linear-layer-enhanced quantum long short-term memory for carbon price forecasting. *Quantum Machine Intelligence*, 5(26), 2023.

[19] Marco Cerezo, Guillaume Verdon, Hsin-Yuan Huang, Lukasz Cincio, and Patrick J. Coles. Challenges and opportunities in quantum machine learning. *Nature Computational Science*, 2(9):567–576, 2022.

[20] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[21] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794. Association for Computing Machinery, 2016.

[22] Elías F. Combarro and Samuel González-Castillo. *A Practical Guide to Quantum Machine Learning and Quantum Optimization: Hands-on Approach to Modern Quantum Algorithms.* Packt Publishing Ltd, Birmingham, 2023.

[23] Iris Cong, Soonwon Choi, and Mikhail D. Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019.

[24] Randall Correll, Sean J. Weinberg, Fabio Sanches, Takanori Ide, and Takafumi Suzuki. Quantum neural networks for a supply chain logistics application. *Advanced Quantum Technologies*, 6(7):2200183, 2023.

[25] Pierre-Luc Dallaire-Demers and Nathan Killoran. Quantum generative adversarial networks. *Physical Review A*, 98(1):012324, 2018.

[26] Adnan Darwiche. Human-level intelligence or animal-like abilities? *Communications of the ACM*, 61(10):56–67, 2018.

[27] Xolani Dastile, Turgay Celik, and Moshe Potsane. Statistical and machine learning models in credit scoring: A systematic literature survey. *Applied Soft Computing*, 91:106263, 2020.

[28] Li Ding and Lee Spector. Evolutionary quantum architecture search for parametrized quantum circuits. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 2190–2195, New York, NY, USA, July 2022. Association for Computing Machinery.

[29] Trong Duong, Sang T. Truong, Minh Pham, Bao Bach, and June-Koo Rhee. Quantum neural architecture search with quantum circuits metric and bayesian optimization. In *ICML 2022 2nd AI for Science Workshop*, June 2022.

[30] Frank Emmert-Streib and Matthias Dehmer. Taxonomy of machine learning paradigms: A data-centric perspective. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 12(5):e1470, 2022.

[31] Eurostat. *NACE Rev. 2 - Statistical classification of economic activities in the European Community.* Eurostat, Luxembourg, 2008. Eurostat Methodologies and Working Papers.

[32] Hadi Fanaee-T. Bike Sharing. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C5W894.

[33] CK-12 Foundation. The polar form of complex numbers. `https://math.libretexts.org/Courses/Rio_Hondo/Math_175%3A_Plane_Trigonometry/06%3A_The_Polar_System/6.04%3A_The_Polar_Form_of_Complex_Numbers`. Accessed: 2024-07-26.

[34] Manoranjan Gandhudi, G. R. Gangadharan, P. J. A. Alphonse, Vasanth Velayudham, and Leeladhar Nagineni. Causal aware parameterized quantum stochastic gradient descent for analyzing marketing advertisements and sales forecasting. *Information Processing & Management*, 60(5):103473, 2023.

[35] Michele Grossi, Noelle Ibrahim, Voica Radescu, Robert Loredo, Kirsten Voigt, Constantin Von Altrock, and Andreas Rudnik. Mixed quantum–classical method for fraud detection with quantum feature selection. *IEEE Transactions on Quantum Engineering*, 3:1–12, 2022.

[36] Lov K. Grover. A fast quantum mechanical algorithm for database search. *arXiv preprint arXiv:quant-ph/9605043v3*, 1996.

[37] Sabrina Herbst, Vincenzo De Maio, and Ivona Brandic. Streaming iot data and the quantum edge: A classic/quantum machine learning use case. *arXiv preprint arXiv:2402.15542v1*, 2024.

[38] Mohamed Hibat-Allah, Marta Mauri, Juan Carrasquilla, and Alejandro Perdomo-Ortiz. A framework for demonstrating practical quantum advantage: comparing quantum against classical generative models. *Communications Physics*, 7(1):68, 2024.

[39] Stefan Hill, David Fitzek, Patrick Delfmann, and Carl Corea. Intercase predictive process monitoring: A candidate for quantum machine learning? *arXiv preprint arXiv:2307.00080*, 2023.

[40] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. *spaCy: Industrial-strength Natural Language Processing in Python*, 2020. `https://doi.org/10.5281/zenodo.1212303`.

[41] Nouhaila Innan, Muhammad Al-Zafar Khan, and Mohamed Bennai. Financial fraud detection: A comparative study of quantum machine learning models. *arXiv preprint arXiv:2308.05237v1*, 2023.

[42] Nouhaila Innan, Abhishek Sawaika, Ashim Dhor, Siddhant Dutta, Sairupa Thota, Husayn Gokal, Nandan Patel, Muhammad Al-Zafar Khan, Ioannis Theodonis, and Mohamed Bennai. Financial fraud detection using quantum graph neural networks. *Quantum Machine Intelligence*, 6(7), 2024.

[43] Md Abrar Jahin, Md Sakib Houssain Shovon, Md Saiful Islam, Jungpil Shin, M. F. Mridha, and Yuichi Okuyama. QAmplifyNet: pushing the boundaries of supply chain backorder prediction using interpretable hybrid quantum-classical neural network. *Scientific Reports*, 13(1):18246, 2023.

[44] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor. *An Introduction to Statistical Learning: With Applications in Python*. Springer Nature, 2023.

[45] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, ..., and Jay M. Gambetta. Quantum computing with qiskit. *arXiv preprint arXiv:2405.08810*, 2024.

[46] Yanjun Ji, Sebastian Brandhofer, and Ilia Polian. Calibration-aware transpilation for variational quantum optimization. In *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 204–214, 2022.

[47] Muhammad Kashif, Muhammad Rashid, Saif Al-Kuwari, and Muhammad Shafique. Alleviating barren plateaus in parameterized quantum machine learning circuits: Investigating advanced parameter initialization strategies. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2024.

[48] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

[49] Youngseok Kim, Andrew Eddins, Sajant Anand, Ken Xuan Wei, Ewout van den Berg, Sami Rosenblatt, Hasan Nayfeh, Yantao Wu, Michael Zaletel, Kristan Temme, and Abhinav Kandala. Evidence for the utility of quantum computing before fault tolerance. *Nature*, 618:500–505, 2023.

[50] David Kremer, Victor Villar, Hanhee Paik, Ivan Duran, Ismael Faro, and Juan Cruz-Benito. Practical and efficient quantum circuit synthesis and transpiling with reinforcement learning. *arXiv preprint arXiv:2405.13196v1*, 2024.

[51] Jatinder Kumar, Deepika Saxena, Ashutosh Kumar Singh, and Athanasios V. Vasilakos. A quantum controlled-not neural network-based load forecast and management model for smart grid. *IEEE Systems Journal*, 2023.

[52] Martin Larocca, Supanut Thanasilp, Samson Wang, Kunal Sharma, Jacob Biamonte, Patrick J. Coles, Lukasz Cincio, Jarrod R. McClean, Zoe Holmes, and M. Cerezo. A review of barren plateaus in variational quantum computing. *arXiv preprint arXiv:2405.00781*, 2024.

[53] IBM Quantum Learning. Variational algorithm design - optimization loops. `https://learning.quantum.ibm.com/course/variational-algorithm-design/optimization-loops`. Accessed: 2024-07-26.

[54] Mingyu Lee, Myeongjin Shin, Junseo Lee, and Kabgyun Jeong. Mutual information maximizing quantum generative adversarial network and its applications in finance. *arXiv preprint arXiv:2309.01363*, 2023.

[55] Frank Leymann, Johanna Barzen, Michael Falkenthal, Daniel Vietz, Benjamin Weder, and Karoline Wild. Quantum in the cloud: application potentials and research opportunities. *arXiv preprint arXiv:2003.06256*, 2020.

[56] Jiaye Li, Jinjing Shi, Jian Zhang, Yuhu Lu, Qin Li, Chunlin Yu, and Shichao Zhang. Quantum nearest neighbor collaborative filtering algorithm for recommendation system. *ACM Transactions on Knowledge Discovery from Data*, 2024.

[57] Jin-Guo Liu and Lei Wang. Differentiable learning of quantum circuit born machines. *Physical Review A*, 98(6):062324, 2018.

[58] Edgar Alonso Lopez-Rojas and Stefan Axelsson. Banksim: A bank payments simulator for fraud detection research. In *26th European Modeling and Simulation Symposium, EMSS*, 2014.

[59] Caio B. S. Maior, Lavínia M. M. Araújo, Isis D. Lins, Márcio Das Chagas Moura, and Enrique López Droguett. Prognostics and health management of rotating machinery via quantum machine learning. *IEEE Access*, 11:25132–25151, 2023.

[60] Alois Mair, Alipasha Vaziri, Gregor Weihs, and Anton Zeilinger. Entanglement of the orbital angular momentum states of photons. *Nature*, 412(6844):313–316, 2001.

[61] Javier Mancilla and Christophe Pere. A preprocessing perspective for quantum machine learning classification advantage in finance using nisq algorithms. *Entropy*, 24(11):1656, 2022.

[62] Javier Mancilla, Andre Sequeira, Tomas Tagliani, Francisco Llaneza, and Claudio Beiza. Empowering credit scoring systems with quantum-enhanced machine learning. *arXiv preprint arXiv:2404.00015*, 2024.

[63] Stephan Matzka. Explainable artificial intelligence for predictive maintenance applications. In *2020 Third International Conference on Artificial Intelligence for Industries (AI4I)*, pages 69–74, Irvine, CA, USA, 2020.

[64] Stephan Matzka. Machine predictive maintenance classification dataset. `https://www.kaggle.com/datasets/shivamb/machine-predictive-maintenance-classification`, 2020. Accessed: 2024-07-26.

[65] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1):4812, 2018.

[66] Piotr Mironowicz, Akshata Shenoy H., Antonio Mandarino, A. Ege Yilmaz, and Thomas Ankenbrand. Applications of quantum machine learning for quantitative finance. *arXiv preprint arXiv:2405.10119v1*, 2024.

[67] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229. Association for Computing Machinery, 2019.

[68] Christoph Molnar. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable.* Lean Publishing, 2020. Available at `https://christophm.github.io/interpretable-ml-book/`.

[69] Maureen Monnet, Nermine Chaabani, Theodora-Augustina Dragan, Balthasar Schachtner, and Jeanette Miriam Lorenz. Understanding

the effects of data encoding on quantum-classical convolutional neural networks. *arXiv preprint arXiv:2405.03027*, 2024.

[70] Marcus R.A. Newman. Bloch sphere. `https://prefetch.eu/know/concept/bloch-sphere`. Accessed: 2024-07-26.

[71] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 10th anniversary edition, 2010.

[72] Cathy O'Neil and Rachel Schutt. *Doing Data Science: Straight Talk from the Frontline*. O'Reilly Media, 2013.

[73] Filippo Orlandi, Enrico Barbierato, and Alice Gatti. Enhancing financial time series prediction with quantum-enhanced synthetic data generation: A case study on the s&p 500 using a quantum wasserstein generative adversarial network approach with a gradient penalty. *Electronics*, 13(11):2158, 2024.

[74] Oumayma Ouedrhiri, Oumayma Banouar, Salah El Hadaj, and Said Raghay. Intelligent recommender system based on quantum clustering and matrix completion. *Concurrency and Computation: Practice and Experience*, 34(15), 2022.

[75] Pallets. Flask (version 3) [computer software], 2024. `https://palletsprojects.com/projects/flask/`.

[76] Eric Paquet and Farzan Soleymani. Quantumleap: Hybrid quantum neural network for financial prediction. *Expert Systems with Applications*, 195:116583, June 2022.

[77] Samir Passi and Solon Barocas. Problem formulation and fairness. In *FAT* '19: Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 39–48. ACM, 2019.

[78] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, ..., and Edouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[79] Koustubh Phalak and Swaroop Ghosh. Qualiti: Quantum machine learning hardware selection for inferencing with top-tier performance. *arXiv preprint arXiv:2405.11194*, 2024.

[80] Marco Pistoia, Syed Farhan Ahmad, Akshay Ajagekar, Alexander Buts, Shouvanik Chakrabarti, Dylan Herman, Shaohan Hu, Andrew Jena, Pierre Minssen, Pradeep Niroula, Arthur Rattew, Yue Sun, and Romina Yalovetzky. Quantum machine learning for finance. In *2021 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2021.

[81] B. Prabadevi, R. Shalini, and B.R. Kavitha. Customer churning analysis using machine learning algorithms. *International Journal of Intelligent Networks*, 4:145–154, 2023.

[82] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79–98, 2018.

[83] Python Software Foundation. *Python Language Reference, version 3.12*, 2024. Available at `https://www.python.org/`.

[84] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2024. `https://www.R-project.org/`.

[85] Carlos A. Riofrio, Oliver Mitevski, Caitlin Jones, Florian Krellner, Aleksandar Vučković, Joseph Doetsch, Johannes Klepsch, Thomas Ehmer, and Andre Luckow. A performance characterization of quantum generative models. *arXiv preprint arXiv:2301.09363v3*, 2024.

[86] Asel Sagingalieva, Stefan Komornyik, Arsenii Senokosov, Ayush Joshi, Alexander Sedykh, Christopher Mansell, Olga Tsurkan, Karan Pinto, Markus Pflitsch, and Alexey Melnikov. Photovoltaic power forecasting using quantum machine learning. *arXiv preprint arXiv:2312.16379v1*, 2023.

[87] Iqbal H. Sarker. Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, 2(6):420, 2021.

[88] Nikolaos Schetakis, Davit Aghamalyan, Michael Boguslavsky, Agnieszka Rees, Marc Rakotomalala, and Paul Robert Griffin. Quantum machine learning for credit scoring. *Mathematics*, 12(9):1391, 2024.

[89] Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. *arXiv preprint arXiv:1803.07128*, 2018.

[90] Maria Schuld and Nathan Killoran. Is quantum advantage the right goal for quantum machine learning? *PRX Quantum*, 3(3):030101, 2022.

[91] Maria Schuld and Francesco Petruccione. *Machine Learning with Quantum Computers*. Springer, Cham, 2nd edition, 2021.

[92] Skipper Seabold and Josef Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010. `https://www.statsmodels.org/`.

[93] Kunal Sharma, M. Cerezo, Zoe Holmes, Lukasz Cincio, Andrew Sornborger, and Patrick J. Coles. Reformulation of the no-free-lunch theorem for entangled datasets. *Physical Review Letters*, 128(7):070501, 2022.

[94] Kevin Shen, Bernhard Jobst, Elvira Shishenina, and Frank Pollmann. Classification of the fashion-mnist dataset on a quantum computer. *arXiv preprint arXiv:2403.02405*, 2024.

[95] Aliza U. Siddiqui, Kaitlin Gili, and Chris Ballance. Stressing out modern quantum hardware: Performance evaluation and execution insights. *arXiv preprint arXiv:2401.13793*, 2024.

[96] Sukin Sim, Peter D. Johnson, and Alan Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technology*, 2, 2019.

[97] James Stokes, Josh Izaac, Nathan Killoran, and Giuseppe Carleo. Quantum natural gradient. *Quantum*, 4:269, 2020.

[98] Horst Stühler, Daniel Pranjic, and Christian Tutschku. Evaluating quantum support vector regression methods for price forecasting applications. In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence (ICAART 2024)*, volume 3, pages 376–384, 2024.

[99] Teppei Suzuki, Takashi Hasebe, and Tsubasa Miyazaki. Quantum support vector machines for classification and regression on a trapped-ion quantum computer. *Quantum Machine Intelligence*, 6(1):31, 2024.

[100] Sohum Thakkar, Skander Kazdaghli, Natansh Mathur, Iordanis Kerenidis, André J. Ferreira-Martins, and Samurai Brito. Improved financial forecasting via quantum machine learning. *Quantum Machine Intelligence*, 6:27, 2024.

[101] Supanut Thanasilp, Samson Wang, Nhat Anh Nghiem, Patrick Coles, and Marco Cerezo. Subtleties in the trainability of quantum machine learning models. *Quantum Machine Intelligence*, 5(1):21, 2023.

[102] Gaurav Topre. Bank customer churn dataset. `https://www.kaggle.com/datasets/gauravtopre/bank-customer-churn-dataset`, 2023. Accessed: 2024-07-26.

[103] Kyriaki A. Tychola, Theofanis Kalampokas, and George A. Papakostas. Quantum machine learning—an overview. *Electronics*, 12(11):2379, 2023.

[104] Lei Wang, Zhengchao Liu, Ang Liu, and Fei Tao. Artificial intelligence in product lifecycle management. *The International Journal of Advanced Manufacturing Technology*, 114:771–796, 2021.

[105] Yunfei Wang and Junyu Liu. A comprehensive review of quantum machine learning: from nisq to fault tolerance. *arXiv preprint arXiv:2401.11351v2*, 2024.

[106] Manuela Weigold, Johanna Barzen, Frank Leymann, and Marie Salm. Encoding patterns for quantum algorithms. *IET Quantum Communication*, 2(4):141–152, 2021.

[107] Manuela Weigold, Johanna Barzen, Frank Leymann, and Marie Salm. Expanding data encoding patterns for quantum algorithms. In *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*, pages 95–101. IEEE, 2021.

[108] Roeland Wiersema, Alexander F. Kemper, Bojko N. Bakalov, and Nathan Killoran. Geometric quantum machine learning with horizontal quantum gates. *arXiv preprint arXiv:2406.04418v1*, 2024.

[109] Rüdiger Wirth and Jochen Hipp. Crisp-dm: Towards a standard process model for data mining. In *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, pages 29–39, 2000.

[110] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:cs.LG/1708.07747*, 2017. Available at `https://github.com/zalandoresearch/fashion-mnist`.

[111] Romina Yalovetzky, Niraj Kumar, Changhao Li, and Marco Pistoia. Qc-forest: a classical-quantum algorithm to provably speedup retraining of random forest. *arXiv preprint arXiv:2406.12008v2*, 2024.

[112] Sheir Yarkoni, Elena Raponi, Thomas Baeck, and Sebastian Schmitt. Quantum annealing for industry applications: introduction and review. *Reports on Progress in Physics*, 85(10):104001, 2022.

[113] I-Cheng Yeh. Default of Credit Card Clients. UCI Machine Learning Repository, 2016. DOI: https://doi.org/10.24432/C55S3H.

Index

A Appendix - Dataset on Student Projects

This appendix contains the dataset that was compiled by the author for the review of Data Science Lab student projects at Vienna University of Economics and Business (see Section 3). Due to non-disclosure requirements, this overview does not contain any company-specific information from the reviewed student reports. Space restrictions also prevent the inclusion of the categories *number of features* and *number of observations*. The full anonymized dataset is provided to the Institute for Data, Process and Knowledge Management at WU Vienna for further improvement of the Data Science Lab course.

Semester	**NACE Code**	**Project Title**	**Business Use Case**	**ML paradigms**	**Methodology**	**Algorithms**	**Tools**	**Perfor-mance Metric**	**Dataset Type**
23W	J6391 - News agency activities	Conversational chatbot for election data platform	Chatbot	NLP / LLM	Text Generation with GPT4 for user prompts to investigate election results; custom UI	Application of pre-trained LLM (GPT)	GPT4; Pandas Langchain Agent; Python; Flask	Accuracy	Text
23W	K6419 - Other monetary intermediation	Website Analytics - User interest and response prediction	Other Marketing Analytics	Supervised Learning, Deep Learning	Predicting goal conversion from URL sequence	LSTM, Entity embedding	Tensorflow / Keras API	Recall	Numerical
23W	M6920 - Accounting, bookkeeping and auditing activities; tax consultancy	Network Analysis for Detecting Shared Risk Factors Between Companies	Valuation	Network analysis	NLP methods (NER. . .) for pre-processing, network visualization / clustering / link prediction for companies and countries	MLP kmeans clustering, Louvain clustering	Python; NetworkX; Gephi; Spacy	na	Graph
23W	J6311 - Data processing, hosting, and related activities	Applying LLMs on Federal Monuments' Text Descriptions	Data Collection and Enrichment	NLP / LLM	Optimizing clarity and standardization of textual monument description	Application of pre-trained LLM (GPT)	GPT4, Gemini, Mistral, Self-deployed Mixtral	Bert Score	Text
23W	J6311 - Data processing, hosting, and related activities	Integration and Accessibility of Open Data in Data Spaces	na	None	Crawling of Open Data Portals and sorting available datasets for clients in Data Space	Elastic NoSQL Search	Python; Elastic-search	na	Public Data APIs

Semester	NACE Code	Project Title	Business Use Case	ML paradigms	Methodology	Algorithms	Tools	Performance Metric	Dataset Type
23W	J6311 - Data processing, hosting, and related activities	Business Process Long Cycle Time Identification and Root Cause Analysis	Process Analytics	Supervised Learning	Binary Classification of overly long activities in order-to-cash process	Jenks Breaks algorithm, Decision Tree	Python; scikit-learn; jenkspy; Neo4j	Accuracy	Graph
23W	H4920 - Freight rail transport	Extraction of identification numbers from freight wagon images	Logistics Automation and Analytics	Vision	Image processing and OCR of textual data	Application of pre-trained LSTM-based OCR	Python; Tesseract; OpenCV	Accuracy	Image
23W	K6419 - Other monetary intermediation	Chatbot for CRM	Chatbot	NLP / LLM	Text Generation with GPT4 for user prompts; custom UI	Application of pre-trained LLM (GPT)	GPT4; Python; Pandas Langchain Agent; tkinter (UI)	Response Time	Text
23W	M7112 - Engineering activities and related technical consultancy	PV-Panel Detection from Aerial Pictures	Infrastructure Planning	Vision	Image pre-processing and segmentation	U-Net (CNN architecture)	Python; scikit-learn; Tensorflow / Keras API	Precision	Image
23W	M7112 - Engineering activities and related technical consultancy	B2B Recommender System	Recommender System	Unsupervised Learning, Recommender Algorithms	Clustering of customers, Association rule mining, collaborative filtering techniques	kMeans clustering, ALS / BPR / WARP algorithms	Python; scikit-learn; implicit; LightFM	AUC	Numerical

Semester	NACE Code	Project Title	Business Use Case	ML paradigms	Methodology	Algorithms	Tools	Perfor-mance Metric	Dataset Type
23W	K6419 - Other monetary intermediation	Fraud model for lending process	Credit Risk Management	Supervised Learning	Predicting potentially fraudulent loan applications, backtesting existing rules	SMOTE, Decision Tree, Logistic Regression	Python; imbalanced-learn; scikit-learn	AUC	Numerical
23W	C2550 - Forging, pressing, stamping, and roll forming of metal; powder metallurgy	Sales forecasting using Multivariate Time Series Models	Demand and Price Prediction	Supervised Learning, Time Series	Predicting future sales values from internal and public data	SARIMAX, Vector Auto-Regression (VAR)	Python; Statsmod-els	MAE	Numerical
23S	J6391 - News agency activities	LLM-based Text Extraction from Natural Language Text	Data Collection and Enrichment	NLP / LLM	Extract election polling data from newspaper articles	Application of pre-trained LLM (GPT)	GPT3.5; GPT4; Python	Accuracy	Text
23S	K6419 - Other monetary intermediation	Client Churn Prediction	Churn Prediction	Supervised Learning	Prediction of client churn from financial data (no demographics)	Decision Tree, Random Forest, Logistic Regression	Python; scikit-learn; imbalanced-learn	Recall	Numerical
23S	M6920 - Accounting, bookkeeping and auditing activities; tax consultancy	Measuring country risk with social media data	Valuation	Supervised Learning, NLP	Estimate risk level of assets based on location via sentiment analysis of social media data	RoBERTa, VADER	Python; transform-ers; vaderSenti-ment	na	Text

Semester	NACE Code	Project Title	Business Use Case	ML paradigms	Methodology	Algorithms	Tools	Perfor-mance Metric	Dataset Type
23S	J6311 - Data processing, hosting, and related activities	Face anonymization	Compliance	Vision	Comparison of different face anonymization methods (bounding boxes, landmarks)	Face pixelation, average and Gaussian blurring	Python; OpenCV	RMSE	Image
23S	M7219 - Other research and experimental development on natural sciences and engineering	Governmental Covid Data Transparency	na	None	Web scraping of data portals and transparency scoring	Transpa-rency scores	Python; R; requests; beautiful-soup	Transpa-rency score	Public Data APIs
23S	M7219 - Other research and experimental development on natural sciences and engineering	Knowledge Graph of Austrian Companies	Data Collection and Enrichment	Knowledge Graphs, NLP	Creation and population of KG with data on relations of Austrian companies	None (web scraping)	Python; Spark; Plotly; Gephi; requests	na	Text
23S	N8299 - Other ancillary business services n.e.c.	Client Churn Prediction	Churn Prediction	Supervised Learning	Comparing models for Customer Lifetime Value-based churn prediction	Decision Tree, Logistic Regression	R	False negatives	Numerical
23S	M7311 - Advertising agencies	Performance analysis of social media posts on Facebook, Instagram, Pinterest	Social Media Analytics	Vision, Supervised Learning	Object detection, image classification, prediction of social media performance	Mulit-task Cascaded CNN	Python; Google Vision	AIC	Image

Semester	NACE Code	Project Title	Business Use Case	ML paradigms	Methodology	Algorithms	Tools	Performance Metric	Dataset Type
23S	K6419 - Other monetary intermediation	Optimizing Prompt Creation for Banking Advertisement Text Generation	Other Marketing Analytics	NLP / LLM	Prompt engineering for ad generation targeted at different customer types	Application of pre-trained LLM (GPT)	GPT3; Python	na	Text
23S	K6419 - Other monetary intermediation	ML for Financial Statements	Data Collection and Enrichment	Vision, Supervised Learning	Improve data extraction from tables and balance sheets in non-uniform financial reports	Application of pre-trained models via tools and libraries (a.o., CNNs)	Python; img2table; Tabulo; Camelot; PDF-Plumber	na	Text
23S	L6832 - Real estate management	Feasibility Study for PV Panels on Campus	Infrastructure Planning	Simulation	Simulate energy generation of PV panels to be installed; assess economic viability	PVWatts Model, Sandia Array Performance Model	Python; PVLIB	na	Numerical
23S	M7022 - Business and other management consulting activities	Vehicle Leasing Price Predictor	Demand and Price Prediction	Supervised Learning	Predicting monthly leasing rate from vehicle details and automotive market data	Decision Tree, Random Forest, XGBoost, AdaBoost Regression, SVR	Python; scikit-learn; XGBoost; Shap	MSE	Numerical

Semester	NACE Code	Project Title	Business Use Case	ML paradigms	Methodology	Algorithms	Tools	Performance Metric	Dataset Type
22W	J6391 - News agency activities	Journalistic Text Automation - Automatic Biography Generation	Data Collection and Enrichment	NLP / LLM	Retrieve biographical information from Wikidata and generate biography texts via LLM prompting	Application of pre-trained LLM (GPT)	Python; SPARQL; Text-Davinci-003	Cosine similarity	Text
22W	K6419 - Other monetary intermediation	Client Churn Prediction	Churn Prediction	Supervised Learning	Prediction of client churn from financial data (no demographics)	Decision Tree, Random Forest, Logistic Regression, Naive Bayes, kNN	Python; scikit-learn	Recall	Numerical
22W	M6920 - Accounting, bookkeeping and auditing activities; tax consultancy	Peer Company Identification based on Business Descriptions	Valuation	NLP	Determine peer groups of companies via similarity metrics of textual descriptions for valuation purposes	TF-IDF, GloVe, BERT	Python; NLTK; scikit-learn; Spacy; transformers	Cosine similarity	Text
22W	M7219 - Other research and experimental development on natural sciences and engineering	Quantifying Green Land Cover of Austrian Municipalities	Research	Vision	Calculating green land cover from geographically masked satellite data	Normalized difference vegetation index	Python; geopandas; shapely; rasterio; OSMnx	na	Image

Semester	**NACE Code**	**Project Title**	**Business Use Case**	**ML paradigms**	**Methodology**	**Algorithms**	**Tools**	**Perfor-mance Metric**	**Dataset Type**
22W	J62 - Computer programming, consultancy, and related activities	Predicting hip implant size from patient data	Demand and Price Prediction	Supervised Learning	Predicting stem- and cup size for total hip arthroplasty to improve hospital inventory management	Gradient Boosting, Random Forest	Python; scikit-learn	RMSE	Numerical
22W	N8299 - Other ancillary business services n.e.c.	Client Churn Prediction	Churn Prediction	Supervised Learning	Churn Prediction based on turnover threshold over time	Decision Tree, Random Forest	R	False negatives	Numerical
22W	H4920 - Freight rail transport	Accelerometer data of rail freight wagons during loading	Logistics Automation and Analytics	Supervised Learning, Deep Learning	Identify factors increasing probability of shocks which might damage wagons during loading	Conv1D for time series	Python; Tensorflow; Shap	Accuracy	Numerical
22W	J6311 - Data processing, hosting, and related activities	(Artificially) Intelligent Investment Decisions using Quantitative Storytelling and Theming	na	None	Building and optimizing theme-based cryptocurrency portfolios and trading strategies	Own optimization implementations of Minimum Variance Portfolio, Sharpe Ratio, and cryptocurrency scores	Python; SciPy	na	Numerical

Semester	NACE Code	Project Title	Business Use Case	ML paradigms	Methodology	Algorithms	Tools	Performance Metric	Dataset Type
22W	K6419 - Other monetary intermediation	Personalized Marketing Content using AI	Customer Segmentation	Unsupervised Learning, Generative Modelling	Cluster customer groups and use text2image model for generation of targeted marketing material	Kmeans clustering, Application of pre-trained text2image model (StableDiffusion)	Python; PyTorch; StableDiffusion; Python Imaging Library	Accuracy	Numerical
22W	K6419 - Other monetary intermediation	Granular Steering of Customer Groups	Customer Segmentation	Supervised Learning, Unsupervised Learning	Identify profitable customer clusters for lending business (segmentation and classification)	DBSCAN, kMeans clustering, Logistic Regression	Python; scikit-learn	AUC	Numerical
22S	J6391 - News agency activities	Event information extraction from emails	Data Collection and Enrichment	NLP	Automated text extraction from emails and provision of events as calendar data	Application of pre-trained Named-Entity-Recognition (SpaCy)	Python; Spacy; prodigy; icalendar	na	Text
22S	K6419 - Other monetary intermediation	Client Satisfaction and Churn Prediction	Churn Prediction	Supervised Learning	Identify impact of portfolio variables on client satisfaction and churn	Linear Regression, Decision Tree, Time-Series Forecasting Models	Python; scikit-learn	MSE	Numerical

Semester	NACE Code	Project Title	Business Use Case	ML paradigms	Methodology	Algorithms	Tools	Perfor-mance Metric	Dataset Type
22S	M6920 - Accounting, bookkeeping and auditing activities; tax consultancy	Peer Company Identification based on Company Filings and Annual Reports	Valuation	NLP	Process documents with NLP tools, implement similarity search function and crossreference results with risk indicators (beta-values)	TF-IDF, BM25	Python; scikit-learn; NLTK; Whoosh; openpyxl; Flask	na	Text
22S	M7219 - Other research and experimental development on natural sciences and engineering	Tracking Extreme Weather Events in Austria	Research	None	Quantification of extreme weather events from historical data and patterns	None (descriptive analysis)	Python; geopandas	na	Numerical
22S	J6311 - Data processing, hosting, and related activities	Modes of Transport - Multiclass Classification	IoT Application	Supervised Learning	Data Augmentation via different synthetic data generation methods and (multiclass) transport mode classification from GPS and sensor data	CTGAN, Tabular VAE, SMOTE, Copula, XGBoost	Python; SDV; imbalanced-learn; Smote-Variants; XGBoost; PyTorch Tabular	TPR	Numerical
22S	J5829 - Other software publishing	Prediction of Ratings, Tags, and Pipelines for Talents in HR Talent Management System	HRM	Supervised Learning	Automated prediction of ratings etc. from professional data on talents available on TMS platform	SMOTE, Random Forest, Naive Bayes, XGBoost	Python; imbalanced-learn; scikit-learn; XGBoost	AUC	Numerical

Semester	NACE Code	Project Title	Business Use Case	ML paradigms	Methodology	Algorithms	Tools	Perfor-mance Metric	Dataset Type
22S	P8559 - Other education n.e.c.	Identification of candidates for coding bootcamp on mobile learning app	Customer Segmentation	Unsupervised Learning	Prediction of suitability for coding bootcamp from user learning paths (event data) on the app	PCA, kMeans Clustering	R; dtplyr; jsonlite; stats	na	Numerical
22S	H4920 - Freight rail transport	Correcting vehicle lists of trains using system status messages enhanced by geospatial data	Logistics Automation and Analytics	None	Code for correcting train lists, followed by geographical and financial analysis of defect wheelsets	None (de-scriptive analysis)	Python; geopy; geopandas;	na	Numerical
22S	J6202 - Computer consultancy activities	Scrap-rate analysis of machines in automotive parts manufacturing	Predictive Maintenance	Supervised Learning	Impact quantification of production anomalies, machine types, shifts, time periods on scrap-rate	SAP AC Classifica-tion, Linear Regression, Logistic Regression	SAP Analytics Cloud; R	MSE	Numerical
22S	K6419 - Other monetary intermediation	Pre-Delinquency Risk Model	Credit Risk Management	Supervised Learning, Unsupervised Learning	Prediction of customers becoming overdue with repayment of lending products, Customer Segmentation	PCA, kMeans, DBSCAN, Hierarchi-cal Clustering, Random Forest	Python; scikit-learn; Optuna; PySpark	AUC	Numerical

Semester	**NACE Code**	**Project Title**	**Business Use Case**	**ML paradigms**	**Methodology**	**Algorithms**	**Tools**	**Perfor-mance Metric**	**Dataset Type**
21W	J6391 - News agency activities	Partially Automated Event Entry	Data Collection and Enrichment	NLP, Supervised Learning	Manually label emails containing events or not, train classifier for event detection	TF-IDF, Naive Bayes, SVM	Python; scikit-learn; NLTK; Spacy	Recall	Text
21W	M6920 - Accounting, bookkeeping and auditing activities; tax consultancy	Generating Portfolios Based on Company Filings and Clustering Shared Risks	Valuation	NLP, Unsupervised Learning	Determine peer groups of companies via similarity of quarterly filings	TF-IDF, MDS, k-means clustering, hierarchi-cal clustering, DBSCAN	Python; scikit-learn	Cosine similar-ity	Text
21W	M6920 - Accounting, bookkeeping and auditing activities; tax consultancy	Modelling Loss Given Default	Credit Risk Management	Supervised Learning	Benchmarking explainable models for LGD prediction on Freddie Mac "Single Family Loan-Level Dataset"	CART Regression Tree, M5 Model Tree, Random Forest, XGBoost	Python; scikit-learn; XGBoost; R; Cubist	RMSE	Numerical
21W	J6311 - Data processing, hosting, and related activities	Fall Detection for Smartwatches and Smartphones	IoT Application	Supervised Learning	Merging publicly available datasets, training predictive models and making them portable for Edge ML application	Random Forest, XGBoost	Python; scikit-learn; R; xgboost; Tensorflow Lite	FPR	Numerical

Semester	**NACE Code**	**Project Title**	**Business Use Case**	**ML paradigms**	**Methodology**	**Algorithms**	**Tools**	**Perfor-mance Metric**	**Dataset Type**
21W	J62 - Computer programming, consultancy, and related activities	Contribution of environmental factors to Covid-19 spread in Vienna	na	None	Replication of a study on quantification of weather effects on Covid spread rates using mechanistic models	SIR Model, Levenberg-Marquardt-algorithm	Python; Scipy	Sum of squared residuals (RSS)	Numerical
21W	H4920 - Freight rail transport	Forecasting train departure and arrival times in freight traffic	Logistics Automation and Analytics	Supervised Learning, Network Analysis	Prediction of freight train delays using ML and spatio-temporal forecasting	Linear Regression, Decision Tree, Random Forest, kNN Regression, SVR, MLP, GCN-LSTM	Python; scikit-learn; Stellar-Graph	R-squared	Numerical
21W	J6202 - Computer consultancy activities	Analysis and interpretation of real-world data in the production environment of an automotive part supplier	Predictive Maintenance	Supervised Learning	Prediction of machine failure ratio from production data using SAP tool	Linear Regression	Python; R; SAP SAC	RMSE	Numerical
21W	K6419 - Other monetary intermediation	Creating an atlas of business clients	Customer Segmentation	Supervised Learning	Prediction whether given SME is customer of the bank to learn about customer characteristics	Random Forest, XGBoost	Python; Scrapy; Plotly; Flask	Accuracy	Numerical

Semester	NACE Code	Project Title	Business Use Case	ML paradigms	Methodology	Algorithms	Tools	Perfor-mance Metric	Dataset Type
21W	K6419 - Other monetary intermediation	Early-Collection Risk Model	Credit Risk Management	Supervised Learning	Segment overdue customers into distinctive clusters (buckets in terms of days-past-due intervals) and predict which customers could move to further buckets	RFECV, LightGBM	Palantir Foundry; Python; lightgbm; PySpark	AUC	Numerical
21W	C2670 - Manufacture of optical instruments	Scalable Matching of Entities Extracted from Multiple Unstructured Datasets	Other Marketing Analytics	Deep Learning, Unsupervised Learning, Supervised Learning	Compare DL algorithms for entity consolidation of CRM data	Auto-EM, Deep-matcher	Python; PyTorch; AutoEM; deep-matcher	F1-Score	Text
21S	J6391 - News agency activities	Classification and Identification of Trending Topics in Social Media	Social Media Analytics	NLP, Supervised Learning	Multi-class classification of Twitter posts into 8 topic categories	TF-IDF, Naive Bayes, SVM	Python; Spacy; NLTK	Accuracy	Text
21S	M6920 - Accounting, bookkeeping and auditing activities; tax consultancy	Build Your Own Synthetic Data Generator	Data Collection and Enrichment	Deep Learning, Unsupervised Learning, Generative Modelling	Exploring Deep Generative Modelling for generating Synthetic Data using 12 public datasets	VAE, CTGAN	Python; scikit-learn; Tensorflow / Keras API; PyTorch; CTGAN	Accuracy	Numerical

Semester	NACE Code	Project Title	Business Use Case	ML paradigms	Methodology	Algorithms	Tools	Perfor-mance Metric	Dataset Type
21S	J6311 - Data processing, hosting, and related activities	Fall Detection	IoT Application	Supervised Learning	Merging publicly available datasets and training fall prediction models	Random Forest, SVM	R; Google AutoML	ROC Curve	Numerical
21S	K6611 - Administration of financial markets	Automatic Analysis of PRIIP-KIDs (Packaged retail investment and insurance products - Key Information Documents)	na	None	Consolidation of previous projects by updating, optimizing and extending web scraping solutions	None (web scraping)	Python; Selenium; R	na	na
21S	J62 - Computer programming, consultancy, and related activities	Holistic AI Infused Management Dashboard	Demand and Price Prediction	Deep Learning, Supervised Learning, Time Series	Creation of a dashboard for visualizing energy prices and related variables, building a spot-price forecasting model using rolling window approach	LSTM	Python; Plotly Dash; scikit-learn; Tensor-flow/Keras API; Talos	MSE	Numerical
21S	H4920 - Freight rail transport	Generating Geofences around Train Stations Based on Wagon Position Data	Logistics Automation and Analytics	Unsupervised Learning	Assignment of wagons' stationary points to stations to calculate geofences (borders of station areas)	DBSCAN	Python; scikit-learn; alphashape	na	Geodata

Semester	NACE Code	Project Title	Business Use Case	ML paradigms	Methodology	Algorithms	Tools	Perfor-mance Metric	Dataset Type
21S	K6419 - Other monetary intermediation	Early Warning for Credit Risk: Scoring Model with Aggregated Sector-specific Data	Credit Risk Management	Supervised Learning	Flagging corporate customers that are likely to default	na	Python	na	Numerical
21S	C2670 - Manufacture of optical instruments	Identification of Relevant Target Groups in Life Science Research using Web Scraping and Semantic Metadata	Customer Segmentation	Unsupervised Learning	Building a knowledge graph about researchers from scraped abstracts and performing topic modelling on abstracts	Latent Dirichlet Allocation	Python; NetworkX; PyViz; NLTK; gensim; SQL; Power BI	na	Text
20W	J6391 - News agency activities	Multiclass Classification of incoming police reports by relevancy	Data Collection and Enrichment	NLP, Supervised Learning	Processing of police reports via bag-of-words with euclidian distance and prediction of relevancy (yes, no, manual inspection needed)	Logistic Regression, Random Forest, Naive Bayes, MLP	Python; NLTK; scikit-learn; Tensor-flow/Keras API	Accuracy	Text
20W	M6920 - Accounting, bookkeeping and auditing activities; tax consultancy	Analyzing broad spectrum company data using synthetic datasets	Compliance	Deep Learning, Unsupervised Learning	Testing synthetization algorithms on internal employee data for GDPR-compliant exploratory analysis	CTGAN, Copula-GAN, TVAE-Synthesizer	R; Synthpop; Python; Synthetic Data Vault	Correlation matrix	Numerical

Semester	NACE Code	Project Title	Business Use Case	ML paradigms	Methodology	Algorithms	Tools	Perfor-mance Metric	Dataset Type
20W	J6311 - Data processing, hosting, and related activities	Mobilio - App User Analytics	Churn Prediction	Supervised Learning	Prediction of client churn from event data of a mobile app detecting phone usage while driving	Decision Tree	R; caret; drake	F1-Score	Numerical
20W	K6611 - Administration of financial markets	Crawling and Matching PRIIP-KIDs (Packaged retail investment and insurance products - Key Information Documents)	na	None	Building scraper for KIDs and implementing a function for checking presence of mandatory KID features	None (web scraping)	R; Rselenium	na	na
20W	J62 - Computer programming, consultancy, and related activities	Day-ahead Price Forecast	Demand and Price Prediction	Time Series	Scraping of public spot price data (ENTSO-E) and price prediction with time series models	Prophet, ARIMA, SARIMAX	Python; Plotly; statsmod-els, fbprophet	RMSE	Numerical
20W	H4920 - Freight rail transport	Tracking Data of Freight Wagons	Logistics Automation and Analytics	Unsupervised Learning	Clustering of wagon positions based on frequency and occurrence of problems	DBSCAN	Python; scikit-learn; Folium	na	Numerical
20W	K6419 - Other monetary intermediation	Automation of Financial Reporting	Reporting	NLP / LLM	Comparison of LLMs for generation of (template-based) financial reports	Ngrams, LSTM, GPT2	Python; NLTK; Tensor-flow/Keras API; PyTorch	Holistic Judg-ment	Text

Semester	**NACE Code**	**Project Title**	**Business Use Case**	**ML paradigms**	**Methodology**	**Algorithms**	**Tools**	**Performance Metric**	**Dataset Type**
20W	K6419 - Other monetary intermediation	Forecasting Business Performance	Credit Risk Management	Supervised Learning	Backtesting data from past (non-ML) forecasts and building a new ML model	Decision Tree, Bayesian Ridge Regression	Python; statsmodels; scikit-learn	R-squared	Numerical
20W	L6832 - Real estate management	Energy Consumption at WU Campus	Infrastructure Planning	Supervised Learning, Time Series	Prediction of consumption factors (heating, water. . .) from historical data	Linear Regression, Prophet, SARIMAX	Python; statsmodels; fbprophet	R-squared	Numerical
20W	C2670 - Manufacture of optical instruments	Predicting purchasing likelihood	Customer Segmentation	Unsupervised Learning	Clustering of dentists via multiple binary variables (services offered) for suggestion of new potential (similar) dentist customers	kMeans clustering, CLARA clustering, PAM clustering, SOM	R; stats; cluster	na	Numerical
20S	J6391 - News agency activities	Processing of incoming police reports	Data Collection and Enrichment	Semi-supervised Learning, Unsupervised Learning	Training of classifier on short-messages for pre-filtering of police reports as relevant or irrelevant	Naive Bayes, Latent Dirichlet Allocation	Python; scikit-learn	Positives, negatives	Text

Semester	**NACE Code**	**Project Title**	**Business Use Case**	**ML paradigms**	**Methodology**	**Algorithms**	**Tools**	**Perfor-mance Metric**	**Dataset Type**
20S	K6611 - Administration of financial markets	Automated Analysis of PRIIP-KIDs (Packaged retail investment and insurance products - Key Information Documents)	Reporting	NLP, Supervised Learning	Web crawling; Multi-class classification of KIDs to calculate complexity indicator from predicted binary labels; building automated reports for results	BERT	Python; Selenium; scikit-learn; simple-transformers; R; Rmark-down	Accuracy	Text
20S	K6419 - Other monetary intermediation	Object Detection using Satellite Imagery	Other Marketing Analytics	Vision, Deep Learning, Supervised Learning	Pool detection and calculation of house sizes from satellite images	YOLOv3	Python; PyTorch	mAP (mean average preci-sion)	Image
20S	K6419 - Other monetary intermediation	Signature Verification	Compliance	Vision, Deep Learning, Supervised Learning	Distinguishing original from forged signatures	SVM, CNN, Image similarity indicators	Python; scikit-image; openCV; Tensorflow / Keras API	False Re-jection Rate	Image
20S	J6311 - Data processing, hosting, and related activities	Covid-19 Emotion Analysis	Research	NLP / LLM, Semi-supervised Learning	Labelling dataset of archived tweets based on emojis and multi-class classification of emotions	Bidirectional LSTM with Attention	Python; Tensorflow / Keras API; NLTK; PySpark	Accuracy	Text

Semester	NACE Code	Project Title	Business Use Case	ML paradigms	Methodology	Algorithms	Tools	Perfor-mance Metric	Dataset Type
20S	L6832 - Real estate management	Energy Consumption at WU Campus	Infrastructure Planning	Time Series, Deep Learning	Seasonality tests and forecasting of heating, cooling, and other energy consumption metrics; building a GUI prototype	Holt-Winters method, ARIMA, SARIMA, LSTM	Python; Plotly; statsmod-els	MAPE (Mean Absolute Percent-age Error)	Numerical
19W	J6391 - News agency activities	Trend Detection on Twitter using Predictive Modelling	Social Media Analytics	Supervised Learning	Web scraping of tweets and prediction of future tweet performance (#likes and #retweets)	Linear Regression	Python; scikit-learn	R-squared	Text
19W	M6920 - Accounting, bookkeeping and auditing activities; tax consultancy	Practical Issues in Business Valuation	Valuation	NLP, Unsupervised Learning	Finding company peer groups for beta calculation via financial data and textual descriptions; building simple GUI	TF-IDF, Hierarchi-cal Clustering, Dynamic Time Warping	Python; scikit-learn; fastdtw; NLTK; Flask	na	Numerical, Text
19W	O 8413 - Regulation of and contribution to more efficient operation of businesses	Legal Thesaurus	UX Improvement	NLP, Unsupervised Learning	Identifying synonyms for terms in RIS search logs by using word embedding distances of pre-trained RIS word2vec model	word2vec	Python; gensim	Recall	Text

Semester	**NACE Code**	**Project Title**	**Business Use Case**	**ML paradigms**	**Methodology**	**Algorithms**	**Tools**	**Perfor-mance Metric**	**Dataset Type**
19W	M7219 - Other research and experimental development on natural sciences and engineering	Geolocating Twitter Users	na	None	Generate coordinates, country and city labels for dataset of twitter users, accessible in virtual machine	None (web scraping)	Python; GeoPy; Geocoder; Dask; Nominatim	na	Text
19W	K6411 - Central banking	Alternative forecast model for Balance of Payments data time series	Data Collection and Enrichment	Time Series	Imputation of missing values of company transaction for Balance of Payments via forecasting	Facebook Prophet	Python; R	MAE	Numerical
19W	K6419 - Other monetary intermediation	Automated Newsletter Creation	Data Collection and Enrichment	Unsupervised Learning	Web scraping of financial news articles, clustering them by similarity and arranging relevant information automatically in newsletter	Latent Dirichlet Allocation	Python; feedparser; newspa-per3k; NLTK; scikit-learn; Flask	Perplexity	Text

Semester	NACE Code	Project Title	Business Use Case	ML paradigms	Methodology	Algorithms	Tools	Perfor-mance Metric	Dataset Type
19W	J5829 - Other software publishing	Knowledge Graph Creation Using the PoolParty Tool Suite	Data Collection and Enrichment	NLP, Knowledge Graphs	Creation of a knowledge discovery / search solution for internal usage of (project, customer, employee) data, combining symbolic and subsymbolic approaches	Knowledge Graphs	Python; PoolParty	na	Linked Data
19W	K6419 - Other monetary intermediation	Predictive Maintenance for Self-Service Devices	Predictive Maintenance	Supervised Learning	Predicting failure of self-service terminals from historical logs and error indicators	na	R; SQL	na	Text
19S	J6391 - News agency activities	Predictive Analytics of Media Topics	Other Marketing Analytics	NLP, Supervised Learning	Predicting magnitude of potential indicator (connections of agency article to host online/print articles) from media topic tags	Linear Regression, Random Forest	Python	R-squared	Text
19S	J62 - Computer programming, consultancy, and related activities	Building a MEL-Code Classification Model	Medical	NLP, Supervised Learning	Suggestion of MEL-codes from covertexts of treatments via TF-IDF information values of keywords	TF-IDF	Python; Spacy; NLTK; heapq	Accuracy	Text

Semester	**NACE Code**	**Project Title**	**Business Use Case**	**ML paradigms**	**Methodology**	**Algorithms**	**Tools**	**Perfor-mance Metric**	**Dataset Type**
19S	Q8610 - Hospital activities	Improved EuroSCORE Report	Medical	Supervised Learning	Prediction of percentage mortality risk of patients undergoing cardiac surgery	Logistic Regression, Naive Bayes, Decision Tree, Random Forest, XGBoost	Python; scikit-learn; xgboost	AUC	Numerical
19S	K6419 - Other monetary intermediation	Transaction Analytics for Insights on Prospects and Possible Payment Defaults	Credit Risk Management	Supervised Learning, Unsupervised Learning	Classification of entities as retail or corporate from transaction data	kMeans Clustering, DBSCAN, Decision Tree, Random Forest	Python; scikit-learn	Accuracy	Numerical
18W	M7022 - Business and other management consulting activities	Finding solutions in large and controversial online deliberations / discussions	Social Media Analytics	NLP	Word embedding of discussion threads and training of a sentiment analysis classifier	word2vec, SVM	Python; NLTK; Spacy; textblob; gensim; scikit-learn	Accuracy	Text
18W	J61 - Telecommu-nications	Smart Traffic Lights	Research	Supervised Learning, Time Series	Prediction of Air Quality Index from weather and traffic data by geographical location in Vienna	Linear Regression	Python; genson; statsmod-els; scikit-learn; geopy; Apache Kafka	R-squared	Numerical

Semester	**NACE Code**	**Project Title**	**Business Use Case**	**ML paradigms**	**Methodology**	**Algorithms**	**Tools**	**Perfor-mance Metric**	**Dataset Type**
18W	J6311 - Data processing, hosting, and related activities	Use cases of Smartphone GPS data	IoT Application	None	Calculating users' home addresses and attended university; comparing GPS data collection to Google's practices	None (de-scriptive analysis)	Python; geopy; Nominatim	na	Geodata
18W	K6419 - Other monetary intermediation	Development and Improvement of a Chatbot	Chatbot	NLP, Deep Learning, Supervised Learning	Building a customer support chatbot with customized components of the open source Rasa Stack (ML for intent and entity classification)	SVM, MLP, LSTM	Python; Rasa Stack; Spacy; scikit-learn; Tensorflow / Keras API	F1-Score	Text
18W	J5829 - Other software publishing	Analysis of Austrian Newspaper Reporting	Social Media Analytics	NLP	Comparison of reporting styles/sentiment in articles and tweets on political parties and individual politicians across tabloids and broadsheets	TF-IDF, PoolParty sentiment analysis	Python; feedparser; tweepy; NLTK; textblob; PoolParty	na	Text
18W	K6419 - Other monetary intermediation	Balance Sheet Analysis	Credit Risk Management	NLP, Vision	Automatically extract positions out of financial statements (pdf) for consequent credit risk analysis	Logistic Regression	pdf2xml; Python; scikit-learn	Accuracy	Image

Semester	**NACE Code**	**Project Title**	**Business Use Case**	**ML paradigms**	**Methodology**	**Algorithms**	**Tools**	**Perfor-mance Metric**	**Dataset Type**
17W	R9101 - Library, archives activities	Data Integration and Open Data in the Context of Bibliographic Data at WU	na	None	Identifying redundancies and combining data sources, identifying Open Data API flaws	None (web scraping)	Python	na	Text
17S	J62 - Computer programming, consultancy, and related activities	Data Science in Healthcare	Medical	Supervised Learning	Predicting and comparing hospital treatment efficiency and success across US states	Linear Regression, Data En-velopment Analysis	R; rDEA	R-squared	Numerical
17S	M7112 - Engineering activities and related technical consultancy	Data Science Strategy	na	None	Data Science Strategy for online B2B platform, including use cases (dynamic pricing, customer segmentation) and legal perspective (data protection)	None	None	na	na

B Appendix - Code for QML Examples

The complete code, data, and documentation for the practical QML implementation work on two proxy cases can be retrieved from the public GitHub repository at `https://github.com/hzeindl/qml-business-applications` under the MIT license.